Model Selection in Estimation of Fitness Landscapes

By
Charles J. Geyer and Ruth G. Shaw

Technical Report No. 671

School of Statistics
University of Minnesota

July 10, 2008

Abstract

A solution to the problem of estimating fitness landscapes was proposed by Lande and Arnold (1983). Another solution, which avoids problematic aspects of the Lande-Arnold methodology, was proposed by Shaw, Geyer, Wagenius, Hangelbroek, and Etterson (2008), who also provided an illustrative example involving real data. An earlier technical report (Geyer and Shaw, 2008) gave an example that was simpler in some ways (the data are simulated from the aster model so there are no issues making the data fit the model one has with real data) and much more complicated in others (each individual has five measured components of fitness over four time periods, 20 variables in all) and illustrates the full richness possible in aster analysis of fitness landscapes. The one issue that technical report did not deal with is model selection. When many phenotypic variables are measured, one often does not know which to put in the model. Lande and Arnold (1983) proposed using principal components regression as a method of dimension reduction, but this method is known to have no theoretical basis. Much of late 20th century and 21st century statistics is about model selection and model averaging, and we apply some of this methodology (which does have strong theoretical basis) to estimation of fitness landscapes using another simulated data set.

All analyses are done in R (R Development Core Team, 2008) using the `aster` contributed package described by Geyer, Wagenius and Shaw (2007) except for analyses in the style of Lande and Arnold (1983), which use ordinary least squares regression. Furthermore, all analyses are done using the `Sweave` function in R, so this entire technical report and all of the analyses reported in it are completely reproducible by anyone who has R with the `aster` package installed and the R noweb file specifying the document.

1 R Package Aster

We use R statistical computing environment (R Development Core Team, 2008) in our analysis. It is free software and can be obtained from `http://cran.r-project.org`. Precompiled binaries are available for Windows, Macintosh, and popular Linux distributions. We use the contributed package `aster`. If R has been installed, but this package has not yet been installed, do

```
install.packages("aster")
```

from the R command line (or do the equivalent using the GUI menus if on Apple Macintosh or Microsoft Windows). This may require root or administrator privileges.

Assuming the `aster` package has been installed, we load it

```
> library(aster)
```

The version of the package used to make this document is 0.7-5 (which is available on CRAN). The version of R used to make this document is 2.7.1.

This entire document and all of the calculations shown were made using the R command `Sweave` and hence are exactly reproducible by anyone who has R and the R noweb (RNW) file from which it was created. Both the RNW file and and the PDF document produced from it are available at `http://www.stat.umn.edu/geyer/aster`. For further details on the use of Sweave and R see Chapter 1 of the technical report by Shaw, et al. (2007) available at the same web site.

Not only can one exactly reproduce the results in the printable document, one can also modify the parameters of the simulation and get different results. Obvious modifications to try are noted on pages 3, 4, 11, and 15 below. But, of course, anything at all can be changed once one has the RNW file.

2 Simulate Data from Conditional Model

2.1 Overview

It is hard to make up an unconditional aster model because the unconditional parameterization is so unintuitive. Thus we proceed in two steps.

- First we simulate data using a conditional aster model with parameters we understand.
- Then we fit an unconditional aster model to these data, and simulate data using the fitted unconditional model.

Also it is hard to make up a fitness landscape, because we model it on the canonical parameter scale and these have no simple connection to the mean value parameter scale. Thus we also proceed in two steps here.

- First we simulate data having a flat fitness landscape.
- Then we fit models having a non-flat fitness landscapes, which we adjust in strength to be moderately statistically significant.

The way these two issues interleave is as follows.

1. We simulate data using a conditional model having a flat fitness landscape.
2. We simulate data using an unconditional model having a flat fitness landscape.
3. We simulate data using unconditional models having a non-flat fitness landscapes.

In Section 2 we only do the first step.

2.2 Graphical Model

We make a graphical model based on consecutive time periods (call them years for concreteness). Data are "observed" over 10 years. For each individual we observe in year i three random variables U_i, V_i, and W_i. All random variables on one individual are connected by the graph shown in Figure 1. The variables U_i and V_i are Bernoulli (zero-or-one valued); the variables W_i are nonnegative-integer-valued. The V_i are redundant: $V_i = 1$ if and only if $W_i > 0$. The variables U_i indicate survival through the i-th year; the variables W_i count offspring produced in the i-th year. Hence the variables V_i indicate successful reproduction in the i-year (one or more offspring). We model the W_i as zero-truncated Poisson given V_i.

If one likes to think of it that way, one can ignore V_i and say that W_i is zero-inflated Poisson given U_i, but the aster way to think about this requires three variables U_i, V_i, and W_i because that fits the required exponential family structure.

In these data, the variable $\sum_i W_i$ is considered fitness (total number of offspring over the course of the study, which will include most of the lifespan).

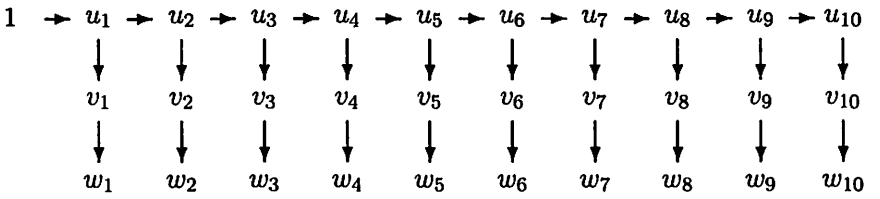

Figure 1: Graph for the example. u_i indicate survival, w_i count number of offspring, v_i indicate $w_i > 0$.

2.3 Simulation

We set the seeds of the random number generator so that we obtain the same results every time. To get different results, obtain the RNW file, change this statement, and reprocess using Sweave and LaTeX.

```
> RNGversion("2.5.0")
> set.seed(42)
```

We also set up some parameters of the simulation. These can also be changed.

```
> nind <- 350
> ntime <- 10
> psurv <- 0.8
> prepr <- 0.7
> mpois <- 3
> theta.surv <- log(psurv) - log(1 - psurv)
> theta.repr <- log(prepr) - log(1 - prepr)
> theta.pois <- log(mpois)
> tau.pois <- mpois/(1 - exp(-mpois))
```

For a first pass we set mean survival per unit time to be psurv $= 0.8$ and mean reproduction indicator per unit time to be prepr $= 0.7$. For number of offspring in a time period given any offspring in that time period, the easiest thing to specify is the mean of the untruncated Poisson random variable that when truncated is the distribution we want. This untruncated mean is mpois $= 3$. The mean of the corresponding zero-truncated distribution is $\tau = 3.157$.

```
> vars <- as.vector(outer(c("u", "v", "w"), 1:10, paste,
+     sep = ""))
> vtype <- as.factor(substr(as.character(vars), 1,
+     1))
> fam <- rep(1, length(vars))
> fam[vtype == "w"] <- 3
> pred <- seq(along = vars) - 1
> pred[vtype == "u"] <- seq(along = vars)[vtype ==
+     "u"] - 3
> pred[1] <- 0
```

Simulate data.

```
> root <- matrix(1, nrow = nind, ncol = length(vars))
> theta <- root
> theta[, vtype == "u"] <- theta.surv
> theta[, vtype == "v"] <- theta.repr
> theta[, vtype == "w"] <- theta.pois
> x <- raster(theta, pred, fam, root)
> dimnames(x) <- list(NULL, vars)
> dat <- as.data.frame(x)
> dat <- as.list(dat)
> dat[["root"]] <- rep(1, nind)
```

Now we add some covariates. These will play the role of phenotypic variables in the Lande-Arnold analysis and the competing aster analysis. Lande-Arnold analysis requires that these be jointly multivariate normal and centered at zero. Aster analysis does not.

For fair comparison, we make these covariates satisfy the conditions required for Lande-Arnold analysis. In addition, we make them have the same variance, and we make all pairs of them have correlation 1/2. The reason for the correlation is that this makes some of the modeling issues more difficult. Of course, all of this can be changed if one has the RNW file for this document.

```
> npheno <- 10
> zbase <- rnorm(nind)
> for (i in 1:npheno) {
+     labz <- paste("z", i, sep = "")
+     dat[[labz]] <- zbase + rnorm(nind)
+ }
```

Reshape the data

```
> names(dat)
```

```
 [1] "u1"   "v1"   "w1"   "u2"   "v2"   "w2"   "u3"   "v3"
 [9] "w3"   "u4"   "v4"   "w4"   "u5"   "v5"   "w5"   "u6"
[17] "v6"   "w6"   "u7"   "v7"   "w7"   "u8"   "v8"   "w8"
[25] "u9"   "v9"   "w9"   "u10"  "v10"  "w10"  "root" "z1"
[33] "z2"   "z3"   "z4"   "z5"   "z6"   "z7"   "z8"   "z9"
[41] "z10"
```

```
> dat <- as.data.frame(dat)
> redata <- reshape(dat, varying = list(vars), direction = "long",
+     timevar = "varb", times = as.factor(vars), v.names = "resp")
```

There is one further step. We wish to make inferences about overall fitness, taking advantage of the monotonicity property of unconditional aster models, as explained in Section 3.2 of Geyer and Shaw (2008). This requires that we model only the W_i as dependent on the phenotypic traits. To accomplish this, we set to zero all the phenotype values except those associated with the W_i, the life history variables that directly reflect fitness.

```
> wind <- grep("w", as.character(redata$varb))
> for (labz in grep("z", names(redata), value = TRUE)) {
+     redata[[labz]][-wind] <- 0
+ }
```

3 Fit Initial Data

3.1 Overview

We have now simulated our "initial data" so we have now done step 1 of the overview in Section 2.1. However, we still do a few checks to see that these data make sense (of course, they must unless the aster software is buggy, but we check anyway); we do this in Section 3.2.

In Section 3.3 we start on what we called step 2 in the overview in Section 2.1. First we fit an unconditional aster model to the data we just checked was reasonable (Section 3.2). Then we "predict" the mean value parameters and see whether they make sense (according to our intuition). They sort of make sense, but we think we can do better, so we fit a more elaborate unconditional model adding a term uyear to the model formula, which lets mortality be a linear function of year on the canonical parameter scale. We again "predict" the mean value parameters and see whether they make sense, and this time decide they are good enough (according to our intuition). That finishes what we called step 2 in the overview in Section 2.1, and also finishes Section 3. We continue our overview in Section 4.

3.2 Conditional Models

Fit the aster model (the one used to generate the data).

```
> redata$vtype <- as.factor(substr(as.character(redata$varb),
+     1, 1))
> out2 <- aster(resp ~ vtype + 0, pred, fam, varb,
+     id, root, data = redata, type = "conditional")
> summary(out2)

Call:
aster.formula(formula = resp ~ vtype + 0, pred = pred, fam = fam,
    varvar = varb, idvar = id, root = root, data = redata, type = "conditional")

       Estimate Std. Error z value Pr(>|z|)
vtypeu  1.36644    0.06323   21.61   <2e-16 ***
vtypev  0.76870    0.06127   12.55   <2e-16 ***
vtypew  1.10173    0.02110   52.22   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

For comparison, these should be

```
> theta.surv
```

```
[1] 1.386294

> theta.repr

[1] 0.8472979

> theta.pois

[1] 1.098612
```

For further comparison, the likelihood ratio test with this model as the alternative hypothesis and the simulation truth for the point null hypothesis can be calculated as follows.

```
> mout2true <- mlogl(c(theta.surv, theta.repr, theta.pois),
+     pred, fam, out2$x, out2$root, out2$modmat, type = out2$type,
+     origin = out2$origin)
> dev2true <- 2 * mout2true$value - out2$deviance
> dev2true

[1] 1.772433

> 1 - pchisq(dev2true, df = 3)

[1] 0.6209522
```

Great! Statistics works. When we simulate data from a model and estimate the parameters in that model, the maximum likelihood estimates are close to the simulation truth, and the log likelihood ratio test statistic comparing the MLE to the simulation truth parameter value is not significant.

3.3 Unconditional Models

The next step is to switch to unconditional models.

```
> out3 <- aster(resp ~ vtype + 0, pred, fam, varb,
+     id, root, data = redata)
> summary(out3)

Call:
aster.formula(formula = resp ~ vtype + 0, pred = pred, fam = fam,
    varvar = varb, idvar = id, root = root, data = redata)

       Estimate Std. Error z value Pr(>|z|)
vtypeu  0.06193    0.04507   1.374    0.169
vtypev -1.64875    0.09005 -18.310   <2e-16 ***
vtypew  1.10173    0.02096  52.560   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We have no intuition about whether this model makes sense. We must switch to unconditional models in order to model fitness, but is this particular unconditional model sensible? To check that out we look at conditional mean value parameters. Do they still make sense? We originally simulated a conditional model, based on certain conditional mean value parameters. Now we have switched to an unconditional model, but we can still calculate its conditional mean value parameters. The new model (`out3`) is "incorrect" in the sense that it is not the old model (`out2`), but is it close to correct in the sense that the relevant parameters have not changed?

In order to compute conditional mean value parameters we need to supply data (since they depend on data, see Geyer, Wagenius and Shaw, 2007, Section 2.5). Since all individuals are alike, we could have a new model matrix with just one individual all of whose variables that are "parent" variables (i. e., all of the U_i and V_i) are equal to one. But a much simpler choice is just to use the observed data, if it contains one such individual.

```
> foo <- apply(out3$x, 1, function(bar) all(bar > 0))
> ifoo <- seq(along = foo)[foo]
> length(ifoo)

[1] 1

> ifoo <- ifoo[1]
> ifoo

[1] 52
```

It does.

```
> pout3 <- predict(out3, model.type = "conditional")
> pout3 <- matrix(pout3, nrow(out3$x))
> pout3 <- pout3[ifoo, ]
> pout3

 [1] 0.8181441 0.6832398 3.1654929 0.8133603 0.6832398 3.1654929
 [7] 0.8073241 0.6832398 3.1654929 0.7996055 0.6832398 3.1654929
[13] 0.7895658 0.6832398 3.1654929 0.7762132 0.6832398 3.1654929
[19] 0.7579195 0.6832398 3.1654929 0.7318098 0.6832398 3.1654929
[25] 0.6922833 0.6832398 3.1654929 0.6267725 0.6832398 3.1654929
```

We see that all of the W_i have conditional expectation 3.1655 given $V_i = 1$. This is "obvious" (we admit we did not guess this in advance, but it is easy to explain in hindsight) from the fact that all W_i are at terminal nodes, hence their conditional canonical parameters θ_j are equal to their unconditional canonical parameters φ_j.

We also see that all of the V_i have conditional expectation 0.6832 given $U_i = 1$. This is "obvious" (we admit we did not guess this in advance, but it is easy to explain in hindsight) from the fact that all V_i have exactly one successor W_i and all of the W_i have the same distribution and the same conditional canonical parameter. Thus equation (5) in Geyer et al. (2007) makes all the unconditional canonical parameters φ_j for the V_i the same.

So we are left with several questions. First, are the conditional mean value parameters already discussed reasonable? They are to be compared with

```
> prepr

[1] 0.7

> tau.pois

[1] 3.157187
```

which are the actual conditional expectations used to simulate the data. They seem close enough. (We know that the difference between 3.1655 and 3.1572 is just sampling variation, because there is no difference between conditional and unconditional canonical parameters for terminal nodes, and the conditional canonical parameters determine the conditional mean value parameters.)

Second, are the conditional mean value parameters for the U_i reasonable? These now all differ, because each has a different number of "ancestor" nodes (predecessor, predecessor of predecessor, etc.)

```
> pout3u <- matrix(pout3, nrow = 3)[1, ]
> round(pout3u, 4)

 [1] 0.8181 0.8134 0.8073 0.7996 0.7896 0.7762 0.7579 0.7318
 [9] 0.6923 0.6268
```

Clearly, our model has changed. Originally, we modeled organisms that do not age: they have the same mortality at all ages. This was just for simplicity; we could have put in mortality that is a function of age. But when we switch to an unconditional model, mortality increases (survival probability decreases) with age.

We have two options. Since we are just making up data, we could accept this change. Alternatively, we could put additional terms into the unconditional aster model to allow the unconditional canonical parameters for the U_i to increase with age more than they do in the model fit above (out3). Let's try the latter.

```
> redata$year <- as.numeric(substring(as.character(redata$varb),
+     2))
> redata$uyear <- redata$year * as.numeric(as.character(redata$vtype) ==
+     "u")
> out4 <- aster(resp ~ vtype + uyear, pred, fam, varb,
+     id, root, data = redata)
> summary(out4)

Call:
aster.formula(formula = resp ~ vtype + uyear, pred = pred, fam = fam,
    varvar = varb, idvar = id, root = root, data = redata)

            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.07095    0.07518  -0.944   0.3453
vtypev      -1.57780    0.13720 -11.500   <2e-16 ***
```

```
vtypew        1.17268    0.07805  15.025   <2e-16 ***
uyear         0.02605    0.01176   2.216   0.0267 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Again we look at conditional mean value parameters.

```
> pout4 <- predict(out4, model.type = "conditional")
> pout4 <- matrix(pout4, nrow(out4$x))
> pout4 <- pout4[ifoo, ]
> pout4
```

```
 [1] 0.7925686 0.6832398 3.1654930 0.8025099 0.6832398 3.1654930
 [7] 0.8094030 0.6832398 3.1654930 0.8128085 0.6832398 3.1654930
[13] 0.8120915 0.6832398 3.1654930 0.8062224 0.6832398 3.1654930
[19] 0.7934028 0.6832398 3.1654930 0.7702688 0.6832398 3.1654930
[25] 0.7299274 0.6832398 3.1654930 0.6561116 0.6832398 3.1654930
```

Again we see that all of the W_i have the same conditional mean value parameter and that it has hardly changed from what it was in out3, and the same is true of all the V_i. Since we did not change the model in any way that affects the parameterization of these variables, this is no surprise.

```
> pout4u <- matrix(pout4, nrow = 3)[1, ]
> round(pout4u, 4)
```

```
 [1] 0.7926 0.8025 0.8094 0.8128 0.8121 0.8062 0.7934 0.7703
 [9] 0.7299 0.6561
```

Now the conditional mean value parameters $E(U_i \mid U_{i-1} = 1)$ for $i > 2$ and $E(U_1)$ seem fairly constant. Perhaps they decrease at large ages (9 and 10), or perhaps this is just chance variation, since there are few individuals alive at these ages. In any event, we will be satisfied with this model out4. This will be our "baseline" model, the model with a flat fitness landscape, since fitness, which we take to be $\sum_i E(W_i)$, does not depend on any of the Z_i, because we put none of these variables in any models fit so far.

4 Simulate New Data

4.1 Overview

At this point we have a reasonable unconditional aster model (out4) with a flat fitness landscape. Now we want to simulate models with non-flat fitness landscapes.

We will simulate two such models, which we call Model 1 and Model 2. In Model 1, the fitness landscape depends on just two phenotypic variables z1 and z2. This is somewhat unrealistic but allows us to draw some simple pictures of fitness landscapes. Burnham and Anderson (2002, Section 1.2.5) are particularly emphatic about the biological unrealism of "true" (at least simulation truth) models with only a few parameters. Hence we also simulate more realistic data that depends on ten phenotypic variables z1, ..., z10.

We make both fitness landscapes depend quadratically on phenotypic variables on the canonical parameter scale. Since the mean value parameter is a multivariate monotone function of the canonical parameter, this means expected fitness is a monotone function of this quadratic function (see Section 3.2 of Geyer and Shaw, 2008 for details). We make our quadratic function have negative curvature (corresponding to stabilizing selection) in all directions. Thus it has elliptical contours. By monotonicity, the contours of the actual fitness landscape (expected fitness considered as a function of phenotypic variables) also has elliptical contours. We locate the fitness maximum to one side of the distribution of phenotypic values not right in the middle.

The only remaining decision is how steeply fitness should fall off as we move away from the fitness maximum. In order for our simulation to provide useful guidance, the fall-off should be steep enough so that the fitness landscape is statistically significant but not so steep that one doesn't need statistics to see that it is significant. (It's always nice when evidence is so clear that statistics is unnecessary, but this is not typical.) Thus we adjust the dependence of fitness on phenotype in both Model 1 and Model 2 to be strong but not too strong.

This proceeds in several steps, which are substeps of what was called Step 3 in Section 2.1.

(a) We fit a model with quadratic fitness landscape to the data with flat fitness landscape. The sole purpose is to get the coefficient vector for such a model.

(b) We adjust the coefficients of the quadratic fitness landscape.

(c) We simulate new data having this unconditional aster model with non-flat (quadratic on the canonical parameter scale) fitness landscape.

(d) We do a likelihood ratio test comparing the models with flat and non-flat fitness landscape. If the result is not statistically significant or is too statistically significant (the inference is so clear that statistics is unnecessary) we go back to substep (b).

4.2 Model 1

For a first try, let us have the fitness landscape depend on just the first two phenotypic variables `z1` and `z2`. Burnham and Anderson (2002, Section 1.2.5) are particularly emphatic about the biological unrealism of "true" (at least simulation truth) models with only a few parameters. The main virtue of this simple model is that it allows us to draw some simple pictures of fitness landscapes. We will also simulate a more complicated model in Section 4.3.

Now we will simulate new data using the `raster` function. To use that we need to, first, make up a model, including dependence on `z1` and `z2` and, second, convert unconditional canonical parameters φ to conditional canonical parameters θ, because that is what the `raster` function wants. There is no function in the `aster` package that does this conversion, perhaps a defect in the `aster` user interface. We can, however, trick the `predict.aster` function into doing this conversion.

First we fit the model we want to use to the data we have. The fitted parameters will make no sense, because the fitness landscape is flat for the data we have, but we can use the model structure.

```
> out5 <- aster(resp ~ vtype + uyear + z1 + z2 + I(z1^2) +
+     I(z2^2) + I(z1 * z2), pred, fam, varb, id, root,
+     data = redata)
> summary(out5)

Call:
aster.formula(formula = resp ~ vtype + uyear + z1 + z2 + I(z1^2) +
    I(z2^2) + I(z1 * z2), pred = pred, fam = fam, varvar = varb,
    idvar = id, root = root, data = redata)

             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.066849   0.075264  -0.888    0.374
vtypev      -1.581175   0.137212 -11.524   <2e-16 ***
vtypew       1.177894   0.078094  15.083   <2e-16 ***
uyear        0.025412   0.011780   2.157    0.031 *
z1          -0.003408   0.005705  -0.597    0.550
z2           0.002567   0.005642   0.455    0.649
I(z1^2)     -0.005393   0.003564  -1.513    0.130
I(z2^2)     -0.003154   0.003514  -0.898    0.369
I(z1 * z2)   0.007144   0.005887   1.213    0.225
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We now want to make up a quadratic function of z that has approximate mean zero (averaged over the z) values so that the average fitness is more or less the same as in our initial data. We also want reasonably sane values, so we keep the quadratic term small. For simplicity, we make the fitness landscape symmetric in the two variables. (It took several tries to get the quadratic term small enough. These tries are not shown. The coefficients of the linear and quadratic terms were adjusted until the P-values in the tests on p. 18 show that both were highly statistically significant but not humongously so.) Of course, all of this can be changed if one has the RNW file for this document.

```
> z1 <- dat$z1
> z2 <- dat$z2
> ascal <- 0.013
> quad <- ascal * ((z1 + z2) - (z1^2 + z2^2) + z1 *
+     z2)
> con <- mean(quad)
> mean(quad - con)

[1] -1.608075e-18
```

Now we change the coefficients in `out5` to be the ones for this made up model.

```
> beta.new <- out5$coefficients
> beta.new[1:4] <- out4$coefficients
> beta.new[3] <- beta.new[3] - con
```

```
> beta.new[5:6] <- ascal
> beta.new[7:8] <- (-ascal)
> beta.new[9] <- ascal
> beta.new <- round(beta.new, 3)
> beta.new

(Intercept)      vtypev      vtypew       uyear          z1
     -0.071      -1.578       1.212       0.026       0.013
         z2     I(z1^2)     I(z2^2)  I(z1 * z2)
      0.013      -0.013      -0.013       0.013
```

Then we "predict" the conditional canonical parameters using `beta.new` because those are the parameters that the `raster` function wants.

```
> theta <- predict(out5, model.type = "conditional",
+     parm.type = "canonical", newcoef = beta.new)
> theta <- matrix(theta, nrow = nrow(out5$x), ncol = ncol(out5$x))
> xnew <- raster(theta, pred, fam, root)
```

Now we need to reshape these new data just like we did the old.

```
> dimnames(xnew) <- list(NULL, vars)
> dnew1 <- as.data.frame(xnew)
> renew <- reshape(dnew1, varying = list(vars), direction = "long",
+     timevar = "varb", times = as.factor(vars), v.names = "resp")
> redata$resp1 <- renew$resp
```

For future reference we also save the unconditional canonical and mean value "simulation truth" parameter values.

```
> redata$tau1 <- predict(out5, newcoef = beta.new)
> redata$phi1 <- predict(out5, parm.type = "canonical",
+     newcoef = beta.new)
> beta1true <- beta.new
```

4.3 Model 2

As pointed out in the Sections 4.1 and 4.2, Burnham and Anderson (2002, Section 1.2.5) are particularly emphatic about the biological unrealism of simulation truth models like that simulated in Section 4.2. In reality, the true stochastic mechanism generating the data is much more complicated than any model under consideration and even if known would have far too many parameters to be well estimated by available data. Not all parameters are equally important, of course, and some parameters of the true model, assuming it were known, could be estimated better than others.

Hence in this section we create a model that is much like the model in Section 4.2 except the quadratic function depends on all of the z_i but not equally.

As before, we start by fitting the full model to the data at hand

```
> out5too <- aster(resp ~ vtype + uyear + poly(z1,
+     z2, z3, z4, z5, z6, z7, z8, z9, z10, degree = 2,
+     raw = TRUE), pred, fam, varb, id, root, data = redata)
> names(out5too$coefficients) <- sub("^poly([^)]*)",
+     "", names(out5too$coefficients), extended = FALSE)
> summary(out5too)

Call:
aster.formula(formula = resp ~ vtype + uyear + poly(z1, z2, z3,
    z4, z5, z6, z7, z8, z9, z10, degree = 2, raw = TRUE), pred = pred,
    fam = fam, varvar = varb, idvar = id, root = root, data = redata)

                     Estimate Std. Error z value Pr(>|z|)
(Intercept)          4.162e-02  7.722e-02   0.539  0.58995
vtypev              -1.666e+00  1.373e-01 -12.136  < 2e-16 ***
vtypew               1.062e+00  8.210e-02  12.938  < 2e-16 ***
uyear                8.635e-03  1.236e-02   0.699  0.48463
1.0.0.0.0.0.0.0.0.0 -6.281e-03  9.411e-03  -0.667  0.50446
2.0.0.0.0.0.0.0.0.0 -1.451e-02  7.002e-03  -2.072  0.03824 *
0.1.0.0.0.0.0.0.0.0  5.151e-03  8.241e-03   0.625  0.53193
1.1.0.0.0.0.0.0.0.0  1.496e-02  1.006e-02   1.487  0.13690
0.2.0.0.0.0.0.0.0.0  1.291e-03  5.302e-03   0.243  0.80764
0.0.1.0.0.0.0.0.0.0  1.404e-02  8.523e-03   1.648  0.09939 .
1.0.1.0.0.0.0.0.0.0 -8.525e-04  9.336e-03  -0.091  0.92724
0.1.1.0.0.0.0.0.0.0  1.425e-02  7.177e-03   1.985  0.04718 *
0.0.2.0.0.0.0.0.0.0  4.609e-03  5.675e-03   0.812  0.41676
0.0.0.1.0.0.0.0.0.0  1.035e-02  7.920e-03   1.307  0.19113
1.0.0.1.0.0.0.0.0.0 -5.166e-03  9.193e-03  -0.562  0.57414
0.1.0.1.0.0.0.0.0.0 -1.719e-03  7.292e-03  -0.236  0.81361
0.0.1.1.0.0.0.0.0.0 -5.085e-03  8.652e-03  -0.588  0.55673
0.0.0.2.0.0.0.0.0.0  5.768e-04  5.198e-03   0.111  0.91164
0.0.0.0.1.0.0.0.0.0 -4.384e-03  7.940e-03  -0.552  0.58085
1.0.0.0.1.0.0.0.0.0  9.889e-05  8.767e-03   0.011  0.99100
0.1.0.0.1.0.0.0.0.0  5.217e-03  7.665e-03   0.681  0.49610
0.0.1.0.1.0.0.0.0.0 -9.425e-03  7.890e-03  -1.195  0.23226
0.0.0.1.1.0.0.0.0.0 -3.258e-03  8.412e-03  -0.387  0.69854
0.0.0.0.2.0.0.0.0.0 -2.384e-03  5.643e-03  -0.422  0.67274
0.0.0.0.0.1.0.0.0.0 -3.488e-03  8.007e-03  -0.436  0.66314
1.0.0.0.0.1.0.0.0.0  4.563e-03  9.204e-03   0.496  0.62005
0.1.0.0.0.1.0.0.0.0 -1.188e-02  7.193e-03  -1.652  0.09856 .
0.0.1.0.0.1.0.0.0.0  4.691e-03  7.994e-03   0.587  0.55729
0.0.0.1.0.1.0.0.0.0  4.473e-03  7.422e-03   0.603  0.54675
0.0.0.0.1.1.0.0.0.0  2.273e-03  7.695e-03   0.295  0.76772
0.0.0.0.0.2.0.0.0.0 -7.602e-03  5.961e-03  -1.275  0.20226
0.0.0.0.0.0.1.0.0.0 -8.375e-04  8.522e-03  -0.098  0.92171
1.0.0.0.0.0.1.0.0.0  9.264e-04  9.080e-03   0.102  0.91873
```

```
0.1.0.0.0.0.1.0.0.0 -1.596e-02  7.586e-03  -2.104  0.03541 *
0.0.1.0.0.0.1.0.0.0 -3.274e-03  8.388e-03  -0.390  0.69632
0.0.0.1.0.0.1.0.0.0  1.276e-02  8.029e-03   1.589  0.11205
0.0.0.0.1.0.1.0.0.0 -1.553e-02  8.900e-03  -1.745  0.08102 .
0.0.0.0.0.1.1.0.0.0 -4.120e-03  8.326e-03  -0.495  0.62074
0.0.0.0.0.0.2.0.0.0  2.921e-04  5.712e-03   0.051  0.95922
0.0.0.0.0.0.0.1.0.0 -2.480e-02  8.684e-03  -2.856  0.00430 **
1.0.0.0.0.0.0.1.0.0  6.809e-03  8.722e-03   0.781  0.43502
0.1.0.0.0.0.0.1.0.0  7.533e-03  7.371e-03   1.022  0.30678
0.0.1.0.0.0.0.1.0.0 -1.924e-02  7.783e-03  -2.472  0.01345 *
0.0.0.1.0.0.0.1.0.0 -5.540e-04  7.493e-03  -0.074  0.94107
0.0.0.0.1.0.0.1.0.0 -4.682e-03  8.259e-03  -0.567  0.57080
0.0.0.0.0.1.0.1.0.0 -7.881e-03  8.907e-03  -0.885  0.37625
0.0.0.0.0.0.1.1.0.0  2.543e-02  9.112e-03   2.790  0.00526 **
0.0.0.0.0.0.0.2.0.0 -1.225e-03  5.913e-03  -0.207  0.83590
0.0.0.0.0.0.0.0.1.0 -2.627e-03  7.811e-03  -0.336  0.73665
1.0.0.0.0.0.0.0.1.0  1.199e-02  9.231e-03   1.299  0.19403
0.1.0.0.0.0.0.0.1.0 -1.823e-02  7.491e-03  -2.434  0.01494 *
0.0.1.0.0.0.0.0.1.0  7.427e-03  7.654e-03   0.970  0.33189
0.0.0.1.0.0.0.0.1.0 -5.882e-03  7.728e-03  -0.761  0.44658
0.0.0.0.1.0.0.0.1.0  8.046e-03  7.374e-03   1.091  0.27522
0.0.0.0.0.1.0.0.1.0  7.262e-03  8.092e-03   0.897  0.36950
0.0.0.0.0.0.1.0.1.0  4.882e-03  7.930e-03   0.616  0.53813
0.0.0.0.0.0.0.1.1.0  6.757e-03  7.330e-03   0.922  0.35667
0.0.0.0.0.0.0.0.2.0 -4.310e-03  5.562e-03  -0.775  0.43845
0.0.0.0.0.0.0.0.0.1  1.394e-02  8.727e-03   1.598  0.11013
1.0.0.0.0.0.0.0.0.1 -1.309e-02  8.734e-03  -1.499  0.13377
0.1.0.0.0.0.0.0.0.1 -8.951e-03  8.082e-03  -1.108  0.26804
0.0.1.0.0.0.0.0.0.1  1.337e-03  8.479e-03   0.158  0.87471
0.0.0.1.0.0.0.0.0.1  4.084e-03  7.476e-03   0.546  0.58487
0.0.0.0.1.0.0.0.0.1  2.158e-02  9.410e-03   2.293  0.02182 *
0.0.0.0.0.1.0.0.0.1  7.224e-03  8.250e-03   0.876  0.38123
0.0.0.0.0.0.1.0.0.1 -1.515e-03  8.600e-03  -0.176  0.86018
0.0.0.0.0.0.0.1.0.1 -2.366e-03  8.054e-03  -0.294  0.76895
0.0.0.0.0.0.0.0.1.1 -3.629e-04  8.603e-03  -0.042  0.96635
0.0.0.0.0.0.0.0.0.2 -7.184e-03  5.963e-03  -1.205  0.22826
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Now we have to figure out which coefficients are which

```
> fred <- strsplit(names(out5too$coefficients)[-(1:4)],
+     "\\.")
> fred <- lapply(fred, as.numeric)
```

Then we construct the quadratic function that is the canonical parameter for fitness. This time we don't bother to adjust the intercept. Of course, all of this can be changed if

one has the RNW file for this document.

```
> b <- 0.5
> bvec <- c(1, 1, b^(1:8))
> bvec <- bvec/sum(bvec) * 2
> round(bvec, 5)

 [1] 0.66754 0.66754 0.33377 0.16688 0.08344 0.04172 0.02086
 [8] 0.01043 0.00522 0.00261
```

Again make a parameter vector `beta.new` constructed above

```
> beta.new <- out5too$coefficients
> beta.new[1:4] <- out4$coefficients
> for (i in seq(along = fred)) {
+     freddy <- fred[[i]]
+     sally <- freddy > 0
+     if (sum(sally) == 1) {
+         if (freddy[sally] == 1) {
+             beta.new[i + 4] <- ascal * bvec[sally]
+         }
+         else {
+             beta.new[i + 4] <- (-ascal * bvec[sally])
+         }
+     }
+     else {
+         beta.new[i + 4] <- ascal * mean(bvec[sally])
+     }
+ }
> beta.new

        (Intercept)              vtypev              vtypew
      -7.095498e-02       -1.577799e+00        1.172683e+00
              uyear 1.0.0.0.0.0.0.0.0.0 2.0.0.0.0.0.0.0.0.0
       2.605057e-02        8.677966e-03       -8.677966e-03
0.1.0.0.0.0.0.0.0.0 1.1.0.0.0.0.0.0.0.0 0.2.0.0.0.0.0.0.0.0
       8.677966e-03        8.677966e-03       -8.677966e-03
0.0.1.0.0.0.0.0.0.0 1.0.1.0.0.0.0.0.0.0 0.1.1.0.0.0.0.0.0.0
       4.338983e-03        6.508475e-03        6.508475e-03
0.0.2.0.0.0.0.0.0.0 0.0.0.1.0.0.0.0.0.0 1.0.0.1.0.0.0.0.0.0
      -4.338983e-03        2.169492e-03        5.423729e-03
0.1.0.1.0.0.0.0.0.0 0.0.1.1.0.0.0.0.0.0 0.0.0.2.0.0.0.0.0.0
       5.423729e-03        3.254237e-03       -2.169492e-03
0.0.0.0.1.0.0.0.0.0 1.0.0.0.1.0.0.0.0.0 0.1.0.0.1.0.0.0.0.0
       1.084746e-03        4.881356e-03        4.881356e-03
0.0.1.0.1.0.0.0.0.0 0.0.0.1.1.0.0.0.0.0 0.0.0.0.2.0.0.0.0.0
       2.711864e-03        1.627119e-03       -1.084746e-03
```

```
0.0.0.0.0.1.0.0.0.0 1.0.0.0.0.1.0.0.0.0 0.1.0.0.0.1.0.0.0.0
       5.423729e-04        4.610169e-03        4.610169e-03
0.0.1.0.0.1.0.0.0.0 0.0.0.1.0.1.0.0.0.0 0.0.0.0.1.1.0.0.0.0
       2.440678e-03        1.355932e-03        8.135593e-04
0.0.0.0.0.2.0.0.0.0 0.0.0.0.0.0.1.0.0.0 1.0.0.0.0.0.1.0.0.0
      -5.423729e-04        2.711864e-04        4.474576e-03
0.1.0.0.0.0.1.0.0.0 0.0.1.0.0.0.1.0.0.0 0.0.0.1.0.0.1.0.0.0
       4.474576e-03        2.305085e-03        1.220339e-03
0.0.0.0.1.0.1.0.0.0 0.0.0.0.0.1.1.0.0.0 0.0.0.0.0.0.2.0.0.0
       6.779661e-04        4.067797e-04       -2.711864e-04
0.0.0.0.0.0.0.1.0.0 1.0.0.0.0.0.0.1.0.0 0.1.0.0.0.0.0.1.0.0
       1.355932e-04        4.406780e-03        4.406780e-03
0.0.1.0.0.0.0.1.0.0 0.0.0.1.0.0.0.1.0.0 0.0.0.0.1.0.0.1.0.0
       2.237288e-03        1.152542e-03        6.101695e-04
0.0.0.0.0.1.0.1.0.0 0.0.0.0.0.0.1.1.0.0 0.0.0.0.0.0.0.2.0.0
       3.389831e-04        2.033898e-04       -1.355932e-04
0.0.0.0.0.0.0.0.1.0 1.0.0.0.0.0.0.0.1.0 0.1.0.0.0.0.0.0.1.0
       6.779661e-05        4.372881e-03        4.372881e-03
0.0.1.0.0.0.0.0.1.0 0.0.0.1.0.0.0.0.1.0 0.0.0.0.1.0.0.0.1.0
       2.203390e-03        1.118644e-03        5.762712e-04
0.0.0.0.0.1.0.0.1.0 0.0.0.0.0.0.1.0.1.0 0.0.0.0.0.0.0.1.1.0
       3.050847e-04        1.694915e-04        1.016949e-04
0.0.0.0.0.0.0.0.2.0 0.0.0.0.0.0.0.0.0.1 1.0.0.0.0.0.0.0.0.1
      -6.779661e-05        3.389831e-05        4.355932e-03
0.1.0.0.0.0.0.0.0.1 0.0.1.0.0.0.0.0.0.1 0.0.0.1.0.0.0.0.0.1
       4.355932e-03        2.186441e-03        1.101695e-03
0.0.0.0.1.0.0.0.0.1 0.0.0.0.0.1.0.0.0.1 0.0.0.0.0.0.1.0.0.1
       5.593220e-04        2.881356e-04        1.525424e-04
0.0.0.0.0.0.0.1.0.1 0.0.0.0.0.0.0.0.1.1 0.0.0.0.0.0.0.0.0.2
       8.474576e-05        5.084746e-05       -3.389831e-05
```

And the rest is just like the Section 4.2.

```
> theta <- predict(out5too, model.type = "conditional",
+     parm.type = "canonical", newcoef = beta.new)
> theta <- matrix(theta, nrow = nrow(out5too$x), ncol = ncol(out5too$x))
> xnew <- raster(theta, pred, fam, root)
```

Reshape.

```
> dimnames(xnew) <- list(NULL, vars)
> dnew2 <- as.data.frame(xnew)
> renew <- reshape(dnew2, varying = list(vars), direction = "long",
+     timevar = "varb", times = as.factor(vars), v.names = "resp")
> redata$resp2 <- renew$resp
```

For future reference we also save the unconditional canonical and mean value "simulation truth" parameter values.

```
> redata$tau2 <- predict(out5too, newcoef = beta.new)
> redata$phi2 <- predict(out5too, parm.type = "canonical",
+     newcoef = beta.new)
> beta2true <- beta.new
```

5 Fit New Data

5.1 Overview

At this point, we have finished making up data (by simulation from an aster model). In real life, none of the work up to here would be necessary. We would collect data by measuring fitness components of real living organisms. For the purposes of this technical report we pretend that the simulated data, `redata$resp` and `redata$resp2` are such real scientific data. Everything we do from here on is an example of what could be done with real data.

5.2 Fit Model 1

So we should be able to fit the model that was used to simulate these data. We start with the simpler model 1 that has only z1 and z2 involved in the simulation truth model.

```
> out6 <- aster(resp1 ~ vtype + uyear + z1 + z2 + I(z1^2) +
+     I(z2^2) + I(z1 * z2), pred, fam, varb, id, root,
+     data = redata)
> summary(out6)

Call:
aster.formula(formula = resp1 ~ vtype + uyear + z1 + z2 + I(z1^2) +
    I(z2^2) + I(z1 * z2), pred = pred, fam = fam, varvar = varb,
    idvar = id, root = root, data = redata)

             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.254846   0.078298  -3.255 0.001135 **
vtypev      -1.352392   0.140791  -9.606  < 2e-16 ***
vtypew       1.423436   0.080121  17.766  < 2e-16 ***
uyear        0.041370   0.011922   3.470 0.000521 ***
z1           0.012508   0.006143   2.036 0.041717 *
z2           0.019975   0.006139   3.254 0.001138 **
I(z1^2)     -0.013450   0.003968  -3.390 0.000699 ***
I(z2^2)     -0.012407   0.003951  -3.141 0.001686 **
I(z1 * z2)   0.010804   0.006250   1.729 0.083862 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

For further comparison, the likelihood ratio test with this model as the alternative hypothesis and the simulation truth for the point null hypothesis can be calculated as follows.

```
> mout6true <- mlogl(beta1true, pred, fam, out6$x,
+     out6$root, out6$modmat, type = out6$type, origin = out6$origin)
> dev6true <- 2 * mout6true$value - out6$deviance
> dev6true

[1] 11.67515

> 1 - pchisq(dev6true, df = length(beta.new))

[1] 1
```

Great! Statistics works again. When we simulate data from a model and estimate the parameters in that model, the maximum likelihood estimates are close to the simulation truth.

We also look at how statistically significant our quadratic effect is

```
> out8 <- aster(resp1 ~ vtype + uyear, pred, fam, varb,
+     id, root, data = redata)
> out7 <- aster(resp1 ~ vtype + uyear + z1 + z2, pred,
+     fam, varb, id, root, data = redata)
> anova(out8, out7, out6)

Analysis of Deviance Table

Model 1: resp1 ~ vtype + uyear
Model 2: resp1 ~ vtype + uyear + z1 + z2
Model 3: resp1 ~ vtype + uyear + z1 + z2 + I(z1^2) + I(z2^2) + I(z1 *
  Model Df Model Dev Df Deviance P(>|Chi|)
1       4   1858.95
2       6   1836.20  2    22.75 1.146e-05
3       9   1810.86  3    25.34 1.312e-05

> anova(out8, out6)

Analysis of Deviance Table

Model 1: resp1 ~ vtype + uyear
Model 2: resp1 ~ vtype + uyear + z1 + z2 + I(z1^2) + I(z2^2) + I(z1 *
  Model Df Model Dev Df Deviance P(>|Chi|)
1       4   1858.95
2       9   1810.86  5    48.09 3.402e-09
```

5.3 Plot Monotone Function of Fitness

We extract the quadratic part of the fitness function (on the canonical parameter scale), which is a monotone transformation of actual fitness, see Sections 3.4, and 3.10 of Shaw, Geyer, Wagenius, Hangelbroek, and Etterson (2007) for details.

```
> qcoef <- out6$coefficients

> a1 <- qcoef["z1"]
> a2 <- qcoef["z2"]
> a <- c(a1, a2)
> A11 <- qcoef["I(z1^2)"]
> A22 <- qcoef["I(z2^2)"]
> A12 <- qcoef["I(z1 * z2)"]/2
> A <- matrix(c(A11, A12, A12, A22), 2, 2)
```

Figure 2 (page 20) shows the scatterplot of data values for z1 and z2 and the contours of the estimated quadratic fitness function. It is similar to Figure 3 in the first submission of Shaw, et al. (2008), which is Figure 3.1 in Shaw, et al. (2007). (In the second submission of the paper, this figure was changed to be like Figure 3 below.) It is made by the following code.

```
> plot(z1, z2)
> ufoo <- par("usr")
> nx <- 101
> ny <- 101
> z <- matrix(NA, nx, ny)
> x <- seq(ufoo[1], ufoo[2], length = nx)
> y <- seq(ufoo[3], ufoo[4], length = ny)
> for (i in 1:nx) {
+     for (j in 1:ny) {
+         b <- c(x[i], y[j])
+         z[i, j] <- sum(a * b) + as.numeric(t(b) %*%
+             A %*% b)
+     }
+ }
> contour(x, y, z, add = TRUE)
```

5.4 Plot Actual Fitness

The plot produced in the preceding section (Figure 2) is good enough for most purposes, even though the numbers on the contours are not actual fitness but a monotone transformation of it (the corresponding canonical parameter). The contours shown are contours of actual fitness, but the numbers attached to them do not reflect actual fitness.

We can get a similar plot where the numbers on the contours are actual fitness. It just takes a bit more work. The first plot of this kind was made in Section 3.11 of Shaw, et al. (2007). Figure 3 of the second submission of Shaw, et al. (2008) was also a plot of this kind.

Figure 3 (page 22) shows the scatterplot of data values for z1 and z2 and the contours of the estimated quadratic fitness function. It is made by the following code.

```
> xx <- outer(x, y^0)
> yy <- outer(x^0, y)
```

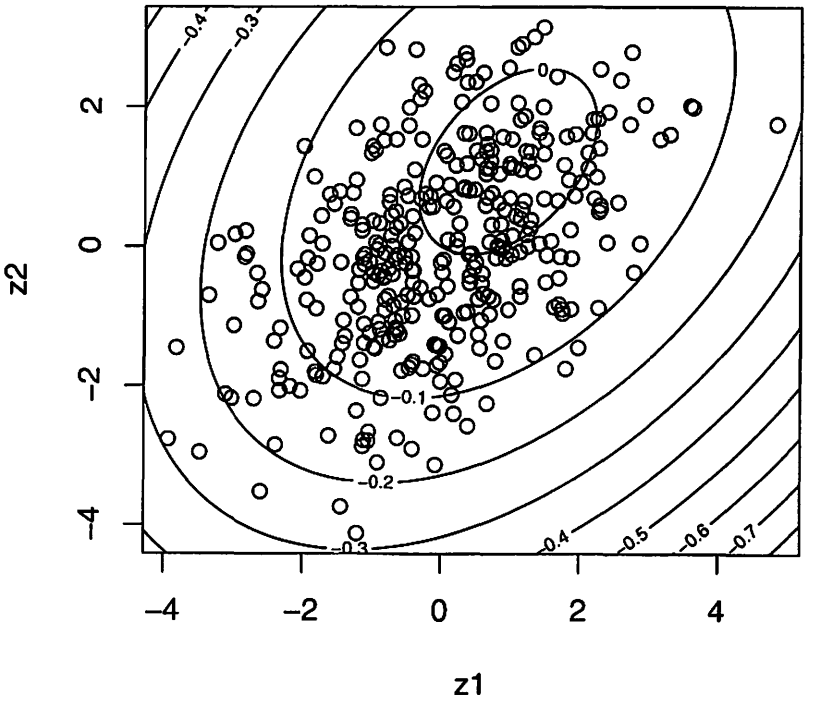

Figure 2: Scatterplot of z1 versus z2 with contours of the estimated quadratic function of phenotypic covariates in the linear predictor, which is only a monotone function of fitness not actual fitness. Compare Figure 3.

```
> xx <- as.vector(xx)
> yy <- as.vector(yy)
> n <- length(xx)
> foo <- rep(1, n)
> bar <- list(z1 = xx, z2 = yy, root = foo)
> for (lab in levels(renew$varb)) {
+     bar[[lab]] <- foo
+ }
> bar <- as.data.frame(bar)
> rebar <- reshape(bar, varying = list(vars), direction = "long",
+     timevar = "varb", times = as.factor(vars), v.names = "resp1")
> rebar$vtype <- as.factor(substr(as.character(rebar$varb),
+     1, 1))
> rebar$year <- as.numeric(substring(as.character(rebar$varb),
+     2))
> rebar$uyear <- rebar$year * as.numeric(as.character(rebar$vtype) ==
+     "u")
> pbar <- predict(out6, newdata = rebar, varvar = varb,
+     idvar = id, root = root)
> pbar <- matrix(pbar, nrow = nrow(bar))
> pbar <- pbar[, grep("w", vars)]
> zz <- apply(pbar, 1, sum)
> zz <- matrix(zz, nx, ny)
```

and

```
> plot(z1, z2)
> contour(x, y, zz, add = TRUE)
```

Figure 3 looks quite a bit different from Figure 2. Nevertheless, every contour of one is also a contour of the other. The only differences are the numbers attached to the contours and which contours the `contour` function draws by default.

6 Lande-Arnold Analysis

6.1 Original

Lande and Arnold (1983) proposed a method of analysis of phenotypic natural selection that is related to the analysis we did leading to Figure 2 but different in the following respects.

First, it uses ordinary least squares (OLS) rather than aster models. Second, it does not estimate the fitness landscape itself. If w is fitness and

$$g(z) = E(w \mid z). \tag{1}$$

is the fitness landscape, define

$$Q(\alpha, \beta, \gamma) = E\{(w - \alpha - z^T\beta - \tfrac{1}{2}z^T\gamma z)^2\} \tag{2}$$

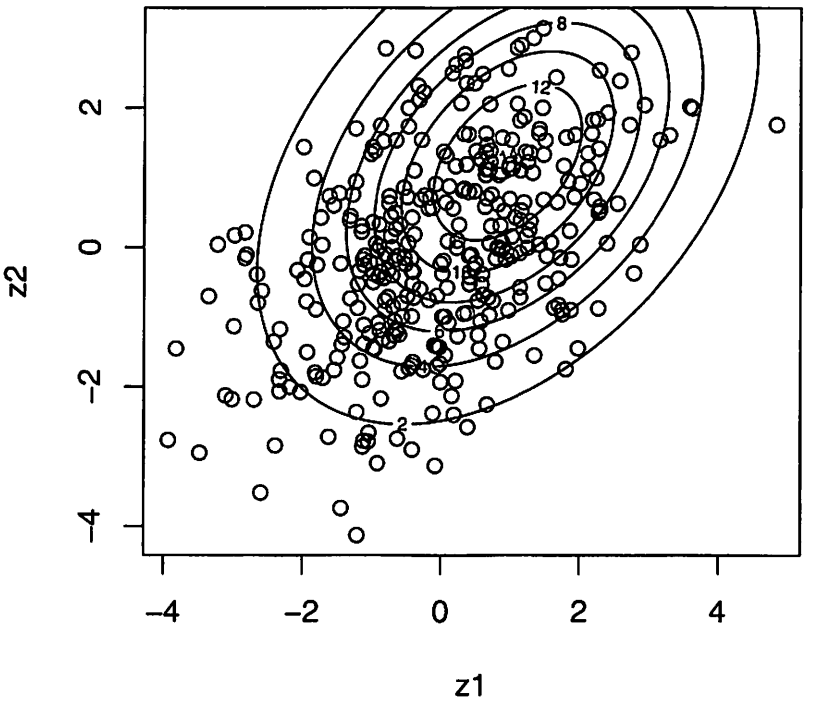

Figure 3: Scatterplot of z1 versus z2 with contours of the estimated fitness function. Note numbers on the contours are actual fitness.

and define α_2, β_2, and γ_2 to be the values of α, β, and γ that minimize $Q(\alpha, \beta, \gamma)$. Then

$$g_2(z) = \alpha_2 + z^T\beta_2 + \tfrac{1}{2}z^T\gamma_2 z \tag{3}$$

the best quadratic approximation to the fitness landscape. Lande and Arnold (1983) show that β_2 and γ_2 have several other equivalent characterizations when z is multivariate normal, but we will have little to say about them in this technical report.

So let us try a Lande-Arnold analysis. First we construct the fitness variable.

```
> widx <- grep("^w[0-9]", dimnames(dnew1)[[2]])
> dnew1$fit <- apply(dnew1[, widx], 1, sum)
```

Then we do the OLS regression that is, in other respects, just like the aster "regression" producing out6.

```
> lout1 <- lm(fit ~ z1 + z2 + I(z1^2) + I(z2^2) + I(z1 *
+     z2), data = dnew1)
> summary(lout1)

Call:
lm(formula = fit ~ z1 + z2 + I(z1^2) + I(z2^2) + I(z1 * z2),
    data = dnew1)

Residuals:
    Min      1Q  Median      3Q     Max
-11.672  -6.950  -2.474   5.992  27.434

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  10.5348     0.7069  14.904  < 2e-16 ***
z1            0.6346     0.3801   1.670 0.095909 .
z2            1.1460     0.3884   2.950 0.003393 **
I(z1^2)      -0.7582     0.2263  -3.351 0.000896 ***
I(z2^2)      -0.6417     0.2389  -2.687 0.007568 **
I(z1 * z2)    0.8290     0.3865   2.145 0.032672 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.923 on 344 degrees of freedom
Multiple R-squared: 0.09935,        Adjusted R-squared: 0.08626
F-statistic: 7.589 on 5 and 344 DF,  p-value: 8.861e-07
```

We now plot the results of the OLS analysis.

```
> qcoef <- lout1$coefficients
> intercept <- qcoef[1]
> a1 <- qcoef["z1"]
> a2 <- qcoef["z2"]
```

```
> a <- c(a1, a2)
> A11 <- qcoef["I(z1^2)"]
> A22 <- qcoef["I(z2^2)"]
> A12 <- qcoef["I(z1 * z2)"]/2
> A <- matrix(c(A11, A12, A12, A22), 2, 2)
```

Figure 4 (page 25) shows the scatterplot of data values for z1 and z2 and the contours of the estimated quadratic function. For comparison, it also shows the contours of the fitness landscape taken from Figure 3. It is made by the following code.

```
> plot(z1, z2)
> contour(x, y, zz, add = TRUE)
> zols <- matrix(NA, nx, ny)
> for (i in 1:nx) {
+     for (j in 1:ny) {
+         b <- c(x[i], y[j])
+         zols[i, j] <- intercept + sum(a * b) + as.numeric(t(b) %*%
+             A %*% b)
+     }
+ }
> contour(x, y, zols, col = "red", add = TRUE)
```

Clearly from Figure 4 the two analyses are qualitatively similar but differ in detail. To the extent they disagree the aster analysis is correct and the Lande-Arnold analysis incorrect (we know this because we know the "true" model under which the data were simulated). The red contours in Figure 4 are the best quadratic approximation to the fitness landscape, but the fitness landscape is not close to quadratic, so even the best quadratic approximation is not a very good approximation. We know fitness is nonnegative, so all of the red contours with negative values are nonsense. To counterbalance the negative nonsense, the best quadratic approximation must underestimate the peak (because it must average to the correct average, a property of best quadratic approximation). The maximum value of the surface with black contours (on the grid where it was evaluated) is 14.1; the maximum value of the surface with red contours (on the same grid) is 12.01.

6.2 Improved by Combination with Aster Analysis

To the extent that we still care about best quadratic approximation now that we know how to estimate the fitness landscape itself, it is interesting that OLS regression is not the best way to estimate β and γ once we have an aster model for the data, which we do (out6).

If we knew the fitness landscape $g(z)$, then we should estimate the best quadratic approximation by regressing $g(z)$ on z rather than regressing w on z. This would give us an estimate that had no statistical error, only error from approximating a non-quadratic function by a quadratic one. We do not know the true fitness landscape, but we do have a good approximation from the aster model, the function whose contours are shown in Figure 3. So we can improve our estimate of the best quadratic approximation to the fitness landscape as follows.

First we go back to the aster analysis (out6) and obtain the unconditional mean value parameters, which in this context are the $E(w \mid z)$.

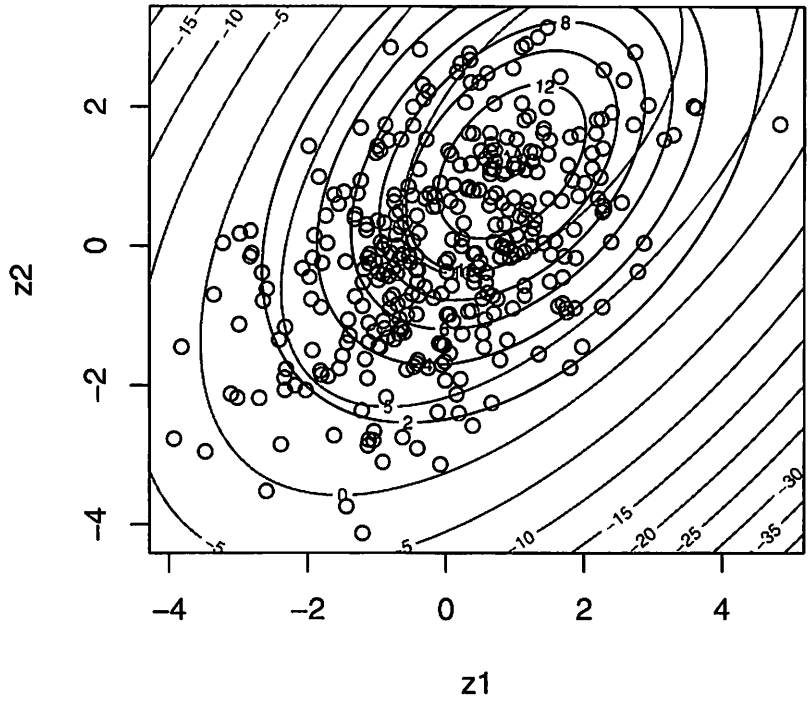

Figure 4: Scatterplot of z1 versus z2 with contours of the estimated fitness landscape (black) and contours of the Lande-Arnold function (red), its best quadratic approximation.

```
> pout6 <- predict(out6)
> pout6 <- matrix(pout6, nrow = nrow(out6$x), ncol = ncol(out6$x))
> wcol <- grep("w", dimnames(out6$x)[[2]])
> afit <- apply(pout6[, wcol], 1, sum)
> dnew1$afit <- afit
```

Then we do the OLS regression that is, in other respects, just like the preceding OLS regression (`lout1`).

```
> lout2 <- lm(afit ~ z1 + z2 + I(z1^2) + I(z2^2) +
+     I(z1 * z2), data = dnew1)
> summary(lout2)

Call:
lm(formula = afit ~ z1 + z2 + I(z1^2) + I(z2^2) + I(z1 * z2),
    data = dnew1)

Residuals:
    Min      1Q  Median      3Q     Max
-2.2666 -0.5600 -0.1239  0.6419  4.2589

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 10.53477    0.06844  153.92   <2e-16 ***
z1           0.63455    0.03680   17.24   <2e-16 ***
z2           1.14596    0.03761   30.47   <2e-16 ***
I(z1^2)     -0.75821    0.02191  -34.60   <2e-16 ***
I(z2^2)     -0.64172    0.02313  -27.75   <2e-16 ***
I(z1 * z2)   0.82902    0.03743   22.15   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8641 on 344 degrees of freedom
Multiple R-squared: 0.9217,        Adjusted R-squared: 0.9205
F-statistic: 809.4 on 5 and 344 DF,  p-value: < 2.2e-16
```

To the extent that the estimates in the two OLS regressions (`lout1` and `lout2`) differ, the latter is better. This is clear from the smallness of the reported standard errors for the latter. These standard errors are invalid, because they are based on the assumption that the response is normal, which it is not (although it is a lot closer to normal in the latter). In order to do the Lande-Arnold analysis, we only needed to assume that z was multivariate normal and needed no assumption about the distribution of w given z. (Of course, we do know that distribution — it is the aster model used to simulate the data — but that distribution does not satisfy the "usual" assumptions for OLS regression.) But invalid or not, the standard errors give a rough indication of the variability of the "response." Of course, correct standard errors could be derived based on the aster model and the delta method, but that is a lot of work.

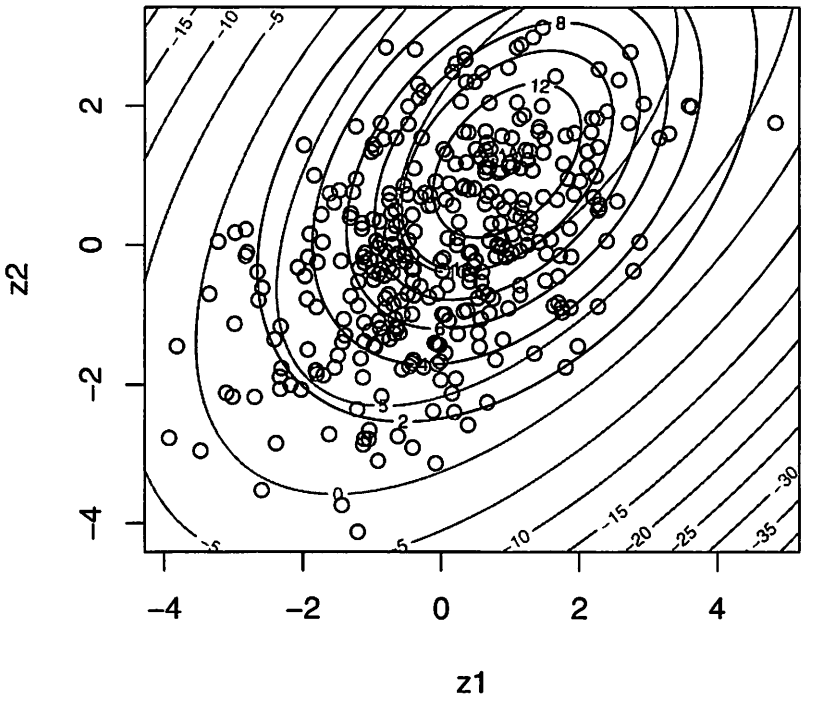

Figure 5: Scatterplot of `z1` versus `z2` with contours of the estimated quadratic fitness function (black) and contours of the Lande-Arnold function (red) using the improved Lande-Arnold estimators.

Figure 5 (page 27) is just like Figure 4 except that now our improved Lande-Arnold estimates based on combining aster analysis with OLS regression (`lout2`) are used instead of the crude OLS estimates.

6.3 Discussion

It is clear from Figure 5 that fitness landscape analysis using aster models and Lande-Arnold analysis estimate different things. The best quadratic approximation to the fitness landscape may not be a very good approximation. It cannot be when the fitness landscape is far from quadratic, which is the usual case. Lande and Arnold (1983) are, of course, not to be faulted for failing to use methods that were first proposed more than 20 years after they wrote their paper, but we can do better now.

7 Model Selection

7.1 Principal Components Regression

Lande and Arnold (1983, p. 1214) have a section on model selection in which they recommend principal components regression (PCR). They are not to be faulted for this because PCR is a very popular methodology that is the subject of many books and articles and has been used for more than 100 years. However, despite its popularity, PCR has no theory that justifies it or even motivates it. Principal components as a method of dimension reduction in multivariate analysis (where there is no response) is justified by theory, but PCR is not. This is well known to statisticians. The problem with PCR is that the principal components (which in this case would be linear combinations of the covariates z_1, z_2, ..., z_{10}) are chosen without any use of the response (in this case w or the entire response vector containing all the u_i, v_i, and w_i). Hence there is no reason why the principal components should be related to the response. PCR may work, but only by fortunate coincidence.

In our first simulated data set (model 1), PCR would give terrible results. We know (because we simulated the data) that z_1 and z_2 are the variables related to w. If we look at the loadings of these variables on the principal components

```
> vz <- dat[, grep("z[0-9]$|z10", names(dat), value = TRUE)]
> names(vz)

 [1] "z1"  "z2"  "z3"  "z4"  "z5"  "z6"  "z7"  "z8"  "z9"  "z10"

> vz <- var(vz)
> ez <- eigen(vz, symmetric = TRUE)
> ez$values

 [1] 10.7165812  1.3526707  1.1623531  1.1181554  1.0792147
 [6]  1.0164283  0.9386040  0.8618710  0.8342036  0.8074655

> ez$vectors[1:2, ]

           [,1]       [,2]        [,3]        [,4]       [,5]
[1,] -0.3285170  0.4792024 -0.07294413 -0.02030048 -0.1486932
[2,] -0.3075170 -0.1992723 -0.23429566 -0.15560890  0.7106091
             [,6]       [,7]       [,8]      [,9]      [,10]
[1,] -0.005553264 0.1129436 -0.3198200 0.3943884 0.6033116
[2,] -0.155499003 0.2673458  0.3453587 0.2328770 0.1117676
```

we see that we need all the principal components to contain the subspace spanned by z_1 and z_2. Thus principal components does exactly the wrong thing in our example. It couldn't be more damaging. Of course, our example was constructed to be this way. In any particular application, principal components may do better (it couldn't do worse), but one can never know when it will do a good job.

In our second simulated data set (model 2), PCR would give slightly less terrible results. The first principal component is now closer to the principal axis of the quadratic form that is

canonical parameter for fitness, but still not close, and the rest of the principal components are random directions having no relationship to fitness.

Cook (2007) gives a thorough discussion of the origins of PCR and what is wrong with it and proposes modern methods that have some of the flavor of PCR without the problems. The earliest criticism of PCR cited by Cook (2007) is Cox (1968), so the problems of PCR have been recognized for at least 40 years.

7.2 Information Criteria

Having given up on the idea that principal components will magically select the correct model, we move to methods that are justified by statistical theory. These methods fit all models in some specified family $\mathcal{M}$ of models, the "models of interest" and evaluate them using some criterion. The oldest and best known criteria are the *Akaike information criterion* (AIC, Akaike, 1973) or the *Bayes information criterion* (BIC, Schwarz, 1978). Many other information criteria have been proposed, but we shall add only one more. Corrected AIC (AICc) which is a finite-sample size correction to AIC, originally due to Sugiura (1978).

If $l(\hat{\theta}_m)$ is the log likelihood maximized over the parameter space of model m and p_m is the dimension of the parameter space of model m, then

$$\text{AIC}(m) = -2l(\hat{\theta}_m) + 2p_m \tag{4}$$

is AIC for model m,

$$\text{AICc}(m) = \text{AIC}(m) + \frac{2p_m[p_m + 1]}{n - p_m - 1} \tag{5}$$

is AICc for model m, and

$$\text{BIC}(m) = -2l(\hat{\theta}_m) + \log(n)p_m \tag{6}$$

is BIC for model m, where n is the sample size. AIC(m) provides a consistent estimator of the maximum expected log likelihood for model m, and BIC(m) provides an asymptotic approximation of the Bayes factor for model m. In our example $\log(n)$ is

```
> log(nind)

[1] 5.857933
```

Note that natural logarithms are used (as they are everywhere in statistics). According to Burnham and Anderson (2002) AICc should always be preferred to AIC unless the sample size is very large.

The idea is that the model with the smallest information criterion, whichever one AIC(m), AICc(m), or BIC(m) we are using, is the best estimate of the true unknown model that we can get. As everywhere else in statistics the estimate is not the truth and is random. When the class $\mathcal{M}$ of models in which we look for the best model is very large, then the probability of selecting the best model may be very small (more on this later). But we shall try these procedures out and see how they do.

7.3 The Branch and Bound Algorithm

We need another idea. Fitting all models in the class $\mathcal{M}$ is unpalatable if the class is large. The *branch and bound* algorithm from computer science was first introduced into statistical model selection by Furnival and Wilson (1974), a paper that is almost unreadable being bogged down in the details of Fortran code for least squares. A good introduction to the branch and bound algorithm is given by Hand (1981).

The branch and bound algorithm works on the principle of divide and conquer. "Branch" refers to dividing up the work to be done. In this case we divide up the work by fixing how one phenotypic covariate enters the model. The canonical parameter is constant, linear, or quadratic in that variable. That divides the whole class of models to be examined into three disjoint subclasses (constant, linear, and quadratic in one particular variable). We can further subdivide the work by splitting on another variable, further subdivide the subdivisions by splitting on yet another, and so forth.

"Bound" refers to bounding the criterion function (e. g., AIC) for a whole subclass of models. If we keep track of the AIC of the best model found so far, this provides an upper bound for the AIC of the best model in the entire class of models to be examined. From (4) we see that, as a function of model complexity, the first term of (4) decreases (a supermodel has higher likelihood than one of its submodels) whereas the second term of (4) increases (a supermodel has more parameters than one of its submodels). Hence a lower bound for AIC of all the models in a subclass $M \subset \mathcal{M}$ of models is

$$-2l_{\text{LCS}(M)}(\hat{\theta}_{\text{LCS}(M)}) + 2p_{\text{GCS}(M)} \tag{7}$$

where $\text{LCS}(M)$ is the least common supermodel (the smallest model that is a supermodel of every m in M) and $\text{GCS}(M)$ is the greatest common submodel (the largest model that is a submodel of every m in M). The same argument provides a lower bound for BIC or AICc because their penalty terms are also increasing functions of p_m. Note further that the lower bound is cheap to evaluate (we only need to fit two models no matter how many models are in M).

If (7) is greater than $\text{AIC}(m)$ for the best model seen so far, then we know that M cannot contain the best model and we need not examine it further. If the lower bound does not allow us to dispense with M, then we subdivide it, finding bounds for each subdivision. When the subdivisions become small enough they will be eliminated (no subdivision can be empty, and if it contains just one model, we evaluate that model and are done with that subdivision).

Branch and bound is not magic. Typically, the number of models to be evaluated grows exponentially in the dimension of the problem. Here we have 3^k models if there are k phenotypic covariates. The branch and bound algorithm still takes time exponential in k, but the constant is much smaller. Typically, the branch and bound algorithm only examines a small fraction of the possible models. The number of models actually examined is reported in each case. So branch and bound does not allow us to do arbitrarily large problems, but it does allow us to do problems somewhat larger than we could do without it.

This algorithm is not implemented in the `aster` package, so we provide an implementation for the particular problem we are interested in here.

```
> source("http://www.stat.umn.edu/geyer/aster/leap-funs.R")
> args(aster.leaps)
```

```
function (pred, fam, data, nsplit, response = "resp", type = c("AIC",
    "AICc", "BIC"), cutoff = 0, envir = new.env(hash = TRUE,
    parent = globalenv()))
NULL
```

The function `aster.leaps` fits all models in a certain class of models, that we now describe. First each model formula starts off "`resp1 ~ vtype + uyear`" just like `out6` if given the argument `response = "resp1"`. The rest of the terms are linear or quadratic in some or all of the variables `z1`, `z2`, ..., the number of such variables being the argument `nsplit` to `aster.leaps`. The linear predictor may be constant, linear, or quadratic in such a variable. If "constant" the variable does not appear in the formula. If "linear" the variable appears linearly, for example, if `z2` and `z3` are linear, then the formula continues "`+ z2 + z3`" If "quadratic" the variable appears quadratically, for example, if `z1`, `z4`, and `z5` are quadratic, then the formula continues "`+ poly(z1, z4, z5, degree = 2, raw = TRUE)`". Note that this means mixed terms of degree two, such as, `z1 * z4` are included in the formula, in effect.

One argument to `aster.leaps`, that is `nsplit`, has been described already. The other non-optional arguments `pred`, `fam`, and `data` are passed to the `aster` function to fit aster models, for this case, we give them the same values they had in producing `out6`, that is `pred`, `fam`, and `redata`.

7.4 Five Predictors

7.4.1 Model 1

Let's try it. We apply the branch and bound algorithm to the data for Model 1 simulated in Section 4.2 in which fitness is a quadratic function of `z1` and `z2` only. Because this function may take a lot of time to run, we store the results in the current working directory, and simply load them if they exist.

```
> options(show.error.messages = FALSE, warn = -1)
> try(load("b1out5a.rda"))
> options(show.error.messages = TRUE, warn = 0)
> if (!exists("b1out5a")) {
+     b1out5a <- aster.leaps(pred, fam, data = redata,
+         nsplit = 5, response = "resp1")
+     save(b1out5a, file = "b1out5a.rda")
+ }
> secs <- b1out5a$time[1]
> mins <- floor(secs/60)
> secs <- floor(secs - mins * 60)
> names(b1out5a)

[1] "fits"  "time"  "envir" "nfit"

> b1out5a$time
```

```
   user  system elapsed
162.882   0.020 172.165

> b1out5a$fits

             dev  p      aic      bic      cic
q1213   1805.962 10 1825.962 1864.541 1826.611
q12135  1805.962 11 1827.962 1870.399 1828.743
q121    1810.862  9 1828.862 1863.583 1829.391
q1214   1809.364 10 1829.364 1867.943 1830.013
q1215   1810.603 10 1830.603 1869.182 1831.252
q12145  1809.303 11 1831.303 1873.740 1832.084
q12513  1805.235 14 1833.235 1887.246 1834.489
q125134 1804.879 15 1834.879 1892.748 1836.316
q1251   1809.784 13 1835.784 1885.937 1836.867
q12514  1808.376 14 1836.376 1890.387 1837.629
q12351  1801.662 18 1837.662 1907.104 1839.728
q123514 1801.245 19 1839.245 1912.546 1841.548
q124513 1804.566 19 1842.566 1915.866 1844.869
q12451  1808.235 18 1844.235 1913.678 1846.302
q145123 1815.911 15 1845.911 1903.780 1847.348
q14512  1819.494 14 1847.494 1901.505 1848.747
q14513  1819.574 14 1847.574 1901.586 1848.828
q24513  1819.576 14 1847.576 1901.587 1848.830
q123451 1799.805 24 1847.805 1940.396 1851.498
q245113 1818.616 15 1848.616 1906.485 1850.053
q2451   1823.638 13 1849.638 1899.791 1850.721
q134512 1811.689 19 1849.689 1922.990 1851.992
q24511  1822.077 14 1850.077 1904.088 1851.331
q1451   1824.718 13 1850.718 1900.871 1851.801
q13451  1815.539 18 1851.539 1920.981 1853.605
q23451  1815.607 18 1851.607 1921.050 1853.674
q34512  1824.161 14 1852.161 1906.172 1853.414
q234511 1814.282 19 1852.282 1925.583 1854.585
q345112 1822.716 15 1852.716 1910.585 1854.153
q34511  1826.884 14 1854.884 1908.895 1856.138
q3451   1829.526 13 1855.526 1905.679 1856.610
```

The call to this function took 2 minutes and 42 seconds. The models fit by the branch and bound algorithm are shown. The deviance, degrees of freedom, AIC, BIC, and AICc (labeled `cic`) are shown for each model. Of the $3^5 = 243$ models under consideration, the branch and bound algorithm fit only `nrow(b1out5a$fits)` $= 31$ in determining the model with the lowest AIC. The labels for the models indicate the degree of each predictor variable. For example, `q12513` means `z1`, `z2`, and `z5` are quadratic, `z3` is linear, and `z4` is constant (not in the model). Note that the simulation truth model ranks number 3 according to AIC of the models that have been fit.

We shall say no more right now about the how well the branch and bound algorithm with AIC worked. Nor will we say anything about performance of similar uses of the branch and bound algorithm with other numbers of predictor variables and other information criteria (AICc and BIC) in the rest of this section. Some aspects of performance will be examined in Section 8.

Now we repeat what we just did using AICc instead of AIC.

```
> options(show.error.messages = FALSE, warn = -1)
> try(load("b1out5c.rda"))
> options(show.error.messages = TRUE, warn = 0)
> if (!exists("b1out5c")) {
+     b1out5c <- aster.leaps(pred, fam, data = redata,
+         nsplit = 5, response = "resp1", type = "AICc",
+         envir = b1out5a$envir)
+     save(b1out5c, file = "b1out5c.rda")
+ }
> secs <- b1out5c$time[1]
> b1out5c$nfit

[1] 0

> identical(dimnames(b1out5a$fits)[[1]], dimnames(b1out5c$fits)[[1]])

[1] FALSE
```

This takes essentially no time (0.012 seconds), since no new fits need to be done. However, the order of models according to AIC and AICc are different. Rather than show all the fits we now show only those having AICc within 10 of the lowest.

```
> ilow <- b1out5c$fits[, "cic"] < b1out5c$fits[1, "cic"] +
+     10
> b1out5c$fits[ilow, , drop = FALSE]

             dev  p      aic      bic      cic
q1213   1805.962 10 1825.962 1864.541 1826.611
q12135  1805.962 11 1827.962 1870.399 1828.743
q121    1810.862  9 1828.862 1863.583 1829.391
q1214   1809.364 10 1829.364 1867.943 1830.013
q1215   1810.603 10 1830.603 1869.182 1831.252
q12145  1809.303 11 1831.303 1873.740 1832.084
q12513  1805.235 14 1833.235 1887.246 1834.489
q125134 1804.879 15 1834.879 1892.748 1836.316
```

The number 10 is arbitrary, but Burnham and Anderson (2002, Section 2.6) make it the dividing line between "considerably less than substantial" support and "essentially none." Note that the simulation truth model ranks number 3 according to AICc of the models that have been fit.

Now we repeat what we just did using BIC instead of AICc.

```
> options(show.error.messages = FALSE, warn = -1)
> try(load("b1out5b.rda"))
> options(show.error.messages = TRUE, warn = 0)
> if (!exists("b1out5b")) {
+     b1out5b <- aster.leaps(pred, fam, data = redata,
+         nsplit = 5, response = "resp1", type = "BIC",
+         envir = b1out5c$envir)
+     save(b1out5b, file = "b1out5b.rda")
+ }
> secs <- b1out5b$time[1]
> mins <- floor(secs/60)
> secs <- floor(secs - mins * 60)
> ilow <- b1out5b$fits[, "bic"] < b1out5b$fits[1, "bic"] +
+     10
> b1out5b$fits[ilow, , drop = FALSE]

            dev  p      aic      bic      cic
q213    1822.091  7 1836.091 1863.097 1836.419
q113    1822.145  7 1836.145 1863.151 1836.473
q112    1822.442  7 1836.442 1863.448 1836.770
q121    1810.862  9 1828.862 1863.583 1829.391
q21     1828.689  6 1840.689 1863.836 1840.933
q1213   1805.962 10 1825.962 1864.541 1826.611
q1214   1809.364 10 1829.364 1867.943 1830.013
q215    1827.604  7 1841.604 1868.609 1841.931
q1135   1821.860  8 1837.860 1868.723 1838.282
q1125   1821.971  8 1837.971 1868.835 1838.394
q2135   1822.046  8 1838.046 1868.910 1838.468
q1215   1810.603 10 1830.603 1869.182 1831.252
q12135  1805.962 11 1827.962 1870.399 1828.743
```

In all three calls to `aster.leaps` we have fit 69 models. In this call we fit 38. The time taken for this call was 1 minutes and 29 seconds. Note that the simulation truth model ranks number 4 according to BIC of the models that have been fit.

7.4.2 Model 2

Now we do the same thing with model 2, simulated in Section 4.3 in which fitness is a quadratic function of `z1`, ..., `z10`.

```
> options(show.error.messages = FALSE, warn = -1)
> try(load("b2out5a.rda"))
> options(show.error.messages = TRUE, warn = 0)
> if (!exists("b2out5a")) {
+     b2out5a <- aster.leaps(pred, fam, data = redata,
+         nsplit = 5, response = "resp2")
+     save(b2out5a, file = "b2out5a.rda")
```

```
+ }
> secs <- b2out5a$time[1]
> mins <- floor(secs/60)
> secs <- floor(secs - mins * 60)
> ilow <- b2out5a$fits[, "aic"] < b2out5a$fits[1, "aic"] +
+     10
> b2out5a$fits[ilow, , drop = FALSE]

            dev  p      aic      bic      cic
q123451 235.9033 24 283.9033 376.4937 287.5956
q12341  248.3135 18 284.3135 353.7563 286.3800
q123415 248.2524 19 286.2524 359.5531 288.5554
```

The call to this function took 1 minutes and 28 seconds. Note that the simulation truth model is not among the models under consideration (it depends on all 10 z_i).

```
> options(show.error.messages = FALSE, warn = -1)
> try(load("b2out5c.rda"))
> options(show.error.messages = TRUE, warn = 0)
> if (!exists("b2out5c")) {
+     b2out5c <- aster.leaps(pred, fam, data = redata,
+         nsplit = 5, response = "resp2", type = "AICc",
+         envir = b2out5a$envir)
+     save(b2out5c, file = "b2out5c.rda")
+ }
> secs <- b2out5c$time[1]
> mins <- floor(secs/60)
> secs <- floor(secs - mins * 60)
> ilow <- b2out5c$fits[, "cic"] < b2out5c$fits[1, "cic"] +
+     10
> b2out5c$fits[ilow, , drop = FALSE]

            dev  p      aic      bic      cic
q12341  248.3135 18 284.3135 353.7563 286.3800
q123451 235.9033 24 283.9033 376.4937 287.5956
q123415 248.2524 19 286.2524 359.5531 288.5554
```

The call to this function took 0 minutes and 0 seconds. Note that the simulation truth model is not among the models under consideration (it depends on all 10 z_i).

```
> options(show.error.messages = FALSE, warn = -1)
> try(load("b2out5b.rda"))
> options(show.error.messages = TRUE, warn = 0)
> if (!exists("b2out5b")) {
+     b2out5b <- aster.leaps(pred, fam, data = redata,
+         nsplit = 5, response = "resp2", type = "BIC",
+         envir = b2out5c$envir)
```

```
+     save(b2out5b, file = "b2out5b.rda")
+ }
> secs <- b2out5b$time[1]
> mins <- floor(secs/60)
> secs <- floor(secs - mins * 60)
> ilow <- b2out5b$fits[, "bic"] < b2out5b$fits[1, "bic"] +
+     10
> b2out5b$fits[ilow, , drop = FALSE]

            dev  p      aic      bic      cic
q12341  248.3135 18 284.3135 353.7563 286.3800
q123415 248.2524 19 286.2524 359.5531 288.5554
```

The time taken for this call was 0 minutes and 46 seconds. Note that the simulation truth model is not among the models under consideration (it depends on all 10 z_i).

7.5 Six Predictors

This section repeats Section 7.4 changing "five" to "six."

7.5.1 Model 1

```
> options(show.error.messages = FALSE, warn = -1)
> try(load("b1out6a.rda"))
> options(show.error.messages = TRUE, warn = 0)
> if (!exists("b1out6a")) {
+     b1out6a <- aster.leaps(pred, fam, data = redata,
+         nsplit = 6, response = "resp1", type = "AIC",
+         envir = b1out5b$envir)
+     save(b1out6a, file = "b1out6a.rda")
+ }
> secs <- b1out6a$time[1]
> mins <- floor(secs/60)
> secs <- floor(secs - mins * 60)
> ilow <- b1out6a$fits[, "aic"] < b1out6a$fits[1, "aic"] +
+     10
> b1out6a$fits[ilow, , drop = FALSE]

             dev  p      aic      bic      cic
q1213    1805.962 10 1825.962 1864.541 1826.611
q12136   1805.133 11 1827.133 1869.570 1827.914
q12134   1805.677 11 1827.677 1870.114 1828.458
q12135   1805.962 11 1827.962 1870.399 1828.743
q121346  1804.540 12 1828.540 1874.836 1829.466
q121     1810.862  9 1828.862 1863.583 1829.391
q1214    1809.364 10 1829.364 1867.943 1830.013
q1215    1810.603 10 1830.603 1869.182 1831.252
```

```
q12145   1809.303 11 1831.303 1873.740 1832.084
q12613   1804.055 14 1832.055 1886.066 1833.308
q12513   1805.235 14 1833.235 1887.246 1834.489
q126134  1803.573 15 1833.573 1891.442 1835.010
q126135  1804.012 15 1834.012 1891.881 1835.449
q125134  1804.879 15 1834.879 1892.748 1836.316
q1261345 1803.560 16 1835.560 1897.287 1837.193
q1261    1809.715 13 1835.715 1885.868 1836.798
q1251    1809.784 13 1835.784 1885.937 1836.867
q12614   1807.797 14 1835.797 1889.808 1837.050
```

The call to this function took 6 minutes and 24 seconds. Note that the simulation truth model ranks number 6 according to AIC of the models that have been fit.

```
> options(show.error.messages = FALSE, warn = -1)
> try(load("b1out6c.rda"))
> options(show.error.messages = TRUE, warn = 0)
> if (!exists("b1out6c")) {
+     b1out6c <- aster.leaps(pred, fam, data = redata,
+         nsplit = 6, response = "resp1", type = "AICc",
+         envir = b1out6a$envir)
+     save(b1out6c, file = "b1out6c.rda")
+ }
> secs <- b1out6c$time[1]
> mins <- floor(secs/60)
> secs <- floor(secs - mins * 60)
> ilow <- b1out6c$fits[, "cic"] < b1out6c$fits[1, "cic"] +
+     10
> b1out6c$fits[ilow, , drop = FALSE]
```

```
            dev  p      aic      bic      cic
q1213   1805.962 10 1825.962 1864.541 1826.611
q12136  1805.133 11 1827.133 1869.570 1827.914
q12134  1805.677 11 1827.677 1870.114 1828.458
q12135  1805.962 11 1827.962 1870.399 1828.743
q121    1810.862  9 1828.862 1863.583 1829.391
q121346 1804.540 12 1828.540 1874.836 1829.466
q1214   1809.364 10 1829.364 1867.943 1830.013
q1215   1810.603 10 1830.603 1869.182 1831.252
q12145  1809.303 11 1831.303 1873.740 1832.084
q12613  1804.055 14 1832.055 1886.066 1833.308
q12513  1805.235 14 1833.235 1887.246 1834.489
q126134 1803.573 15 1833.573 1891.442 1835.010
q126135 1804.012 15 1834.012 1891.881 1835.449
q125134 1804.879 15 1834.879 1892.748 1836.316
q213    1822.091  7 1836.091 1863.097 1836.419
q113    1822.145  7 1836.145 1863.151 1836.473
```

The call to this function took 0 minutes and 0 seconds. Note that the simulation truth model ranks number 5 according to AICc of the models that have been fit.

```
> options(show.error.messages = FALSE, warn = -1)
> try(load("b1out6b.rda"))
> options(show.error.messages = TRUE, warn = 0)
> if (!exists("b1out6b")) {
+     b1out6b <- aster.leaps(pred, fam, data = redata,
+         nsplit = 6, response = "resp1", type = "BIC",
+         envir = b1out6c$envir)
+     save(b1out6b, file = "b1out6b.rda")
+ }
> secs <- b1out6b$time[1]
> mins <- floor(secs/60)
> secs <- floor(secs - mins * 60)
> ilow <- b1out6b$fits[, "bic"] < b1out6b$fits[1, "bic"] +
+     10
> b1out6b$fits[ilow, , drop = FALSE]

           dev  p      aic      bic      cic
q213   1822.091  7 1836.091 1863.097 1836.419
q113   1822.145  7 1836.145 1863.151 1836.473
q112   1822.442  7 1836.442 1863.448 1836.770
q121   1810.862  9 1828.862 1863.583 1829.391
q21    1828.689  6 1840.689 1863.836 1840.933
q1123  1817.493  8 1833.493 1864.356 1833.915
q1213  1805.962 10 1825.962 1864.541 1826.611
q1214  1809.364 10 1829.364 1867.943 1830.013
q215   1827.604  7 1841.604 1868.609 1841.931
q2136  1821.847  8 1837.847 1868.711 1838.270
q1135  1821.860  8 1837.860 1868.723 1838.282
q1125  1821.971  8 1837.971 1868.835 1838.394
q1136  1822.003  8 1838.003 1868.867 1838.426
q2135  1822.046  8 1838.046 1868.910 1838.468
q1126  1822.126  8 1838.126 1868.989 1838.548
q1216  1810.576 10 1830.576 1869.155 1831.225
q1215  1810.603 10 1830.603 1869.182 1831.252
q11236 1816.582  9 1834.582 1869.303 1835.111
q12136 1805.133 11 1827.133 1869.570 1827.914
q216   1828.668  7 1842.668 1869.674 1842.996
q12134 1805.677 11 1827.677 1870.114 1828.458
q12135 1805.962 11 1827.962 1870.399 1828.743
```

The time taken for this call was 5 minutes and 6 seconds. Note that the simulation truth model ranks number 4 according to BIC of the models that have been fit.

7.5.2 Model 2

```
> options(show.error.messages = FALSE, warn = -1)
> try(load("b2out6a.rda"))
> options(show.error.messages = TRUE, warn = 0)
> if (!exists("b2out6a")) {
+     b2out6a <- aster.leaps(pred, fam, data = redata,
+         nsplit = 6, response = "resp2")
+     save(b2out6a, file = "b2out6a.rda")
+ }
> secs <- b2out6a$time[1]
> mins <- floor(secs/60)
> secs <- floor(secs - mins * 60)
> ilow <- b2out6a$fits[, "aic"] < b2out6a$fits[1, "aic"] +
+     10
> b2out6a$fits[ilow, , drop = FALSE]

             dev  p      aic      bic      cic
q1234561 204.5308 31 266.5308 386.1268 272.7698
q123461  218.5492 24 266.5492 359.1396 270.2415
q1234615 218.5485 25 268.5485 364.9968 272.5608
```

The call to this function took 3 minutes and 8 seconds. Note that the simulation truth model is not among the models under consideration (it depends on all 10 z_i).

```
> options(show.error.messages = FALSE, warn = -1)
> try(load("b2out6c.rda"))
> options(show.error.messages = TRUE, warn = 0)
> if (!exists("b2out6c")) {
+     b2out6c <- aster.leaps(pred, fam, data = redata,
+         nsplit = 6, response = "resp2", type = "AICc",
+         envir = b2out6a$envir)
+     save(b2out6c, file = "b2out6c.rda")
+ }
> secs <- b2out6c$time[1]
> mins <- floor(secs/60)
> secs <- floor(secs - mins * 60)
> ilow <- b2out6c$fits[, "cic"] < b2out6c$fits[1, "cic"] +
+     10
> b2out6c$fits[ilow, , drop = FALSE]

             dev  p      aic      bic      cic
q123461  218.5492 24 266.5492 359.1396 270.2415
q1234615 218.5485 25 268.5485 364.9968 272.5608
q1234561 204.5308 31 266.5308 386.1268 272.7698
```

The call to this function took 0 minutes and 13 seconds. Note that the simulation truth model is not among the models under consideration (it depends on all 10 z_i).

```
> options(show.error.messages = FALSE, warn = -1)
> try(load("b2out6b.rda"))
> options(show.error.messages = TRUE, warn = 0)
> if (!exists("b2out6b")) {
+     b2out6b <- aster.leaps(pred, fam, data = redata,
+         nsplit = 6, response = "resp2", type = "BIC",
+         envir = b2out6c$envir)
+     save(b2out6b, file = "b2out6b.rda")
+ }
> secs <- b2out6b$time[1]
> mins <- floor(secs/60)
> secs <- floor(secs - mins * 60)
> ilow <- b2out6b$fits[, "bic"] < b2out6b$fits[1, "bic"] +
+     10
> b2out6b$fits[ilow, , drop = FALSE]

             dev  p      aic      bic      cic
q12341  248.3135 18 284.3135 353.7563 286.3800
q123416 247.7541 19 285.7541 359.0549 288.0572
q123461 218.5492 24 266.5492 359.1396 270.2415
q123415 248.2524 19 286.2524 359.5531 288.5554
```

The time taken for this call was 5 minutes and 33 seconds. Note that the simulation truth model is not among the models under consideration (it depends on all 10 z_i).

7.6 Seven Predictors

This section repeats Section 7.5 changing "six" to "seven."

7.6.1 Model 1

```
> options(show.error.messages = FALSE, warn = -1)
> try(load("b1out7a.rda"))
> options(show.error.messages = TRUE, warn = 0)
> if (!exists("b1out7a")) {
+     b1out7a <- aster.leaps(pred, fam, data = redata,
+         nsplit = 7, response = "resp1", type = "AIC",
+         envir = b1out5b$envir)
+     save(b1out7a, file = "b1out7a.rda")
+ }
> secs <- b1out7a$time[1]
> mins <- floor(secs/60)
> secs <- floor(secs - mins * 60)
> ilow <- b1out7a$fits[, "aic"] < b1out7a$fits[1, "aic"] +
+     10
> b1out7a$fits[ilow, , drop = FALSE]
```

	dev	p	aic	bic	cic
q1213	1805.962	10	1825.962	1864.541	1826.611
q12136	1805.133	11	1827.133	1869.570	1827.914
q12137	1805.439	11	1827.439	1869.877	1828.220
q12134	1805.677	11	1827.677	1870.114	1828.458
q12135	1805.962	11	1827.962	1870.399	1828.743
q12713	1800.006	14	1828.006	1882.017	1829.259
q121346	1804.540	12	1828.540	1874.836	1829.466
q121367	1804.763	12	1828.763	1875.059	1829.689
q121	1810.862	9	1828.862	1863.583	1829.391
q121347	1805.006	12	1829.006	1875.301	1829.932
q127136	1799.042	15	1829.042	1886.911	1830.479
q121356	1805.118	12	1829.118	1875.413	1830.043
q1214	1809.364	10	1829.364	1867.943	1830.013
q121357	1805.424	12	1829.424	1875.719	1830.350
q127134	1799.505	15	1829.505	1887.374	1830.942
q121345	1805.667	12	1829.667	1875.962	1830.593
q12371	1793.907	18	1829.907	1899.349	1831.973
q127135	1800.005	15	1830.005	1887.874	1831.442
q1213467	1804.026	13	1830.026	1880.179	1831.110
q1271346	1798.142	16	1830.142	1891.869	1831.776
q1231	1804.280	13	1830.280	1880.433	1831.363
q12146	1808.539	11	1830.539	1872.976	1831.320
q1213456	1804.540	13	1830.540	1880.693	1831.623
q1216	1810.576	10	1830.576	1869.155	1831.225
q1215	1810.603	10	1830.603	1869.182	1831.252
q1213567	1804.710	13	1830.710	1880.863	1831.793
q1271	1804.718	13	1830.718	1880.871	1831.802
q123716	1792.797	19	1830.797	1904.098	1833.100
q1217	1810.855	10	1830.855	1869.434	1831.504
q1213457	1805.004	13	1831.004	1881.157	1832.088
q1271356	1799.032	16	1831.032	1892.759	1832.666
q12714	1803.051	14	1831.051	1885.062	1832.305
q12147	1809.215	11	1831.215	1873.652	1831.996
q12145	1809.303	11	1831.303	1873.740	1832.084
q12316	1803.376	14	1831.376	1885.387	1832.629
q1271345	1799.490	16	1831.490	1893.217	1833.124
q123714	1793.620	19	1831.620	1904.921	1833.923
q127146	1801.887	15	1831.887	1889.756	1833.324
q12314	1803.888	14	1831.888	1885.899	1833.142
q123715	1793.893	19	1831.893	1905.194	1834.196
q12317	1803.930	14	1831.930	1885.941	1833.183
q12134567	1804.005	14	1832.005	1886.016	1833.259
q12613	1804.055	14	1832.055	1886.066	1833.308
q12713456	1798.142	17	1832.142	1897.727	1833.985

```
q1237146  1792.147 20 1832.147 1909.305 1834.700
q12716    1804.190 14 1832.190 1886.201 1833.443
q12156    1810.191 11 1832.191 1874.629 1832.972
q12315    1804.274 14 1832.274 1886.285 1833.528
q121456   1808.403 12 1832.403 1878.698 1833.328
q121467   1808.457 12 1832.457 1878.752 1833.383
q12157    1810.558 11 1832.558 1874.995 1833.339
q12167    1810.576 11 1832.576 1875.013 1833.357
q12715    1804.585 14 1832.585 1886.596 1833.839
q123146   1802.603 15 1832.603 1890.472 1834.040
q1237156  1792.797 20 1832.797 1909.955 1835.350
q127145   1803.032 15 1833.032 1890.901 1834.469
q121457   1809.105 12 1833.105 1879.400 1834.031
q123167   1803.152 15 1833.152 1891.021 1834.589
q12513    1805.235 14 1833.235 1887.246 1834.489
q123156   1803.370 15 1833.370 1891.239 1834.807
q123147   1803.409 15 1833.409 1891.278 1834.846
q1123     1817.493  8 1833.493 1864.356 1833.915
q126134   1803.573 15 1833.573 1891.442 1835.010
q1237145  1793.584 20 1833.584 1910.743 1836.137
q126137   1803.652 15 1833.652 1891.521 1835.090
q1271456  1801.822 16 1833.822 1895.549 1835.456
q123145   1803.858 15 1833.858 1891.727 1835.296
q123157   1803.928 15 1833.928 1891.797 1835.365
q127156   1803.959 15 1833.959 1891.828 1835.396
q126135   1804.012 15 1834.012 1891.881 1835.449
q12371456 1792.143 21 1834.143 1915.160 1836.960
q121567   1810.175 12 1834.175 1880.470 1835.101
q1231467  1802.261 16 1834.261 1895.988 1835.895
q1214567  1808.271 13 1834.271 1884.424 1835.354
q11236    1816.582  9 1834.582 1869.303 1835.111
q1231456  1802.602 16 1834.602 1896.329 1836.235
q125134   1804.879 15 1834.879 1892.748 1836.316
q126713   1796.911 19 1834.911 1908.211 1837.214
q13712    1806.955 14 1834.955 1888.966 1836.209
q1231567  1803.126 16 1835.126 1896.853 1836.759
q1231457  1803.406 16 1835.406 1897.133 1837.040
q1261345  1803.560 16 1835.560 1897.287 1837.193
q1261     1809.715 13 1835.715 1885.868 1836.798
q1251     1809.784 13 1835.784 1885.937 1836.867
q12614    1807.797 14 1835.797 1889.808 1837.050
```

The call to this function took 29 minutes and 31 seconds. Note that the simulation truth model ranks number 9 according to AIC of the models that have been fit.

```
> options(show.error.messages = FALSE, warn = -1)
> try(load("b1out7c.rda"))
> options(show.error.messages = TRUE, warn = 0)
> if (!exists("b1out7c")) {
+     b1out7c <- aster.leaps(pred, fam, data = redata,
+         nsplit = 7, response = "resp1", type = "AICc",
+         envir = b1out7a$envir)
+     save(b1out7c, file = "b1out7c.rda")
+ }
> secs <- b1out7c$time[1]
> mins <- floor(secs/60)
> secs <- floor(secs - mins * 60)
> ilow <- b1out7c$fits[, "cic"] < b1out7c$fits[1, "cic"] +
+     10
> b1out7c$fits[ilow, , drop = FALSE]

              dev  p      aic      bic      cic
q1213     1805.962 10 1825.962 1864.541 1826.611
q12136    1805.133 11 1827.133 1869.570 1827.914
q12137    1805.439 11 1827.439 1869.877 1828.220
q12134    1805.677 11 1827.677 1870.114 1828.458
q12135    1805.962 11 1827.962 1870.399 1828.743
q12713    1800.006 14 1828.006 1882.017 1829.259
q121      1810.862  9 1828.862 1863.583 1829.391
q121346   1804.540 12 1828.540 1874.836 1829.466
q121367   1804.763 12 1828.763 1875.059 1829.689
q121347   1805.006 12 1829.006 1875.301 1829.932
q1214     1809.364 10 1829.364 1867.943 1830.013
q121356   1805.118 12 1829.118 1875.413 1830.043
q121357   1805.424 12 1829.424 1875.719 1830.350
q127136   1799.042 15 1829.042 1886.911 1830.479
q121345   1805.667 12 1829.667 1875.962 1830.593
q127134   1799.505 15 1829.505 1887.374 1830.942
q1213467  1804.026 13 1830.026 1880.179 1831.110
q1216     1810.576 10 1830.576 1869.155 1831.225
q1215     1810.603 10 1830.603 1869.182 1831.252
q12146    1808.539 11 1830.539 1872.976 1831.320
q1231     1804.280 13 1830.280 1880.433 1831.363
q127135   1800.005 15 1830.005 1887.874 1831.442
q1217     1810.855 10 1830.855 1869.434 1831.504
q1213456  1804.540 13 1830.540 1880.693 1831.623
q1271346  1798.142 16 1830.142 1891.869 1831.776
q1213567  1804.710 13 1830.710 1880.863 1831.793
q1271     1804.718 13 1830.718 1880.871 1831.802
q12371    1793.907 18 1829.907 1899.349 1831.973
q12147    1809.215 11 1831.215 1873.652 1831.996
```

q12145	1809.303	11	1831.303	1873.740	1832.084
q1213457	1805.004	13	1831.004	1881.157	1832.088
q12714	1803.051	14	1831.051	1885.062	1832.305
q12316	1803.376	14	1831.376	1885.387	1832.629
q1271356	1799.032	16	1831.032	1892.759	1832.666
q12156	1810.191	11	1832.191	1874.629	1832.972
q123716	1792.797	19	1830.797	1904.098	1833.100
q1271345	1799.490	16	1831.490	1893.217	1833.124
q12314	1803.888	14	1831.888	1885.899	1833.142
q12317	1803.930	14	1831.930	1885.941	1833.183
q12134567	1804.005	14	1832.005	1886.016	1833.259
q12613	1804.055	14	1832.055	1886.066	1833.308
q127146	1801.887	15	1831.887	1889.756	1833.324
q121456	1808.403	12	1832.403	1878.698	1833.328
q12157	1810.558	11	1832.558	1874.995	1833.339
q12167	1810.576	11	1832.576	1875.013	1833.357
q121467	1808.457	12	1832.457	1878.752	1833.383
q12716	1804.190	14	1832.190	1886.201	1833.443
q12315	1804.274	14	1832.274	1886.285	1833.528
q12715	1804.585	14	1832.585	1886.596	1833.839
q1123	1817.493	8	1833.493	1864.356	1833.915
q123714	1793.620	19	1831.620	1904.921	1833.923
q12713456	1798.142	17	1832.142	1897.727	1833.985
q121457	1809.105	12	1833.105	1879.400	1834.031
q123146	1802.603	15	1832.603	1890.472	1834.040
q123715	1793.893	19	1831.893	1905.194	1834.196
q127145	1803.032	15	1833.032	1890.901	1834.469
q12513	1805.235	14	1833.235	1887.246	1834.489
q123167	1803.152	15	1833.152	1891.021	1834.589
q1237146	1792.147	20	1832.147	1909.305	1834.700
q123156	1803.370	15	1833.370	1891.239	1834.807
q123147	1803.409	15	1833.409	1891.278	1834.846
q126134	1803.573	15	1833.573	1891.442	1835.010
q126137	1803.652	15	1833.652	1891.521	1835.090
q121567	1810.175	12	1834.175	1880.470	1835.101
q11236	1816.582	9	1834.582	1869.303	1835.111
q123145	1803.858	15	1833.858	1891.727	1835.296
q1237156	1792.797	20	1832.797	1909.955	1835.350
q1214567	1808.271	13	1834.271	1884.424	1835.354
q123157	1803.928	15	1833.928	1891.797	1835.365
q127156	1803.959	15	1833.959	1891.828	1835.396
q126135	1804.012	15	1834.012	1891.881	1835.449
q1271456	1801.822	16	1833.822	1895.549	1835.456
q1231467	1802.261	16	1834.261	1895.988	1835.895
q1237145	1793.584	20	1833.584	1910.743	1836.137

```
q13712    1806.955 14 1834.955 1888.966 1836.209
q1231456  1802.602 16 1834.602 1896.329 1836.235
q125134   1804.879 15 1834.879 1892.748 1836.316
q213      1822.091  7 1836.091 1863.097 1836.419
q113      1822.145  7 1836.145 1863.151 1836.473
```

The call to this function took 0 minutes and 0 seconds. Note that the simulation truth model ranks number 7 according to AICc of the models that have been fit.

```
> options(show.error.messages = FALSE, warn = -1)
> try(load("b1out7b.rda"))
> options(show.error.messages = TRUE, warn = 0)
> if (!exists("b1out7b")) {
+     b1out7b <- aster.leaps(pred, fam, data = redata,
+         nsplit = 7, response = "resp1", type = "BIC",
+         envir = b1out7c$envir)
+     save(b1out7b, file = "b1out7b.rda")
+ }
> secs <- b1out7b$time[1]
> mins <- floor(secs/60)
> secs <- floor(secs - mins * 60)
> ilow <- b1out7b$fits[, "bic"] < b1out7b$fits[1, "bic"] +
+     10
> b1out7b$fits[ilow, , drop = FALSE]

            dev  p      aic      bic      cic
q213   1822.091  7 1836.091 1863.097 1836.419
q113   1822.145  7 1836.145 1863.151 1836.473
q112   1822.442  7 1836.442 1863.448 1836.770
q121   1810.862  9 1828.862 1863.583 1829.391
q21    1828.689  6 1840.689 1863.836 1840.933
q1123  1817.493  8 1833.493 1864.356 1833.915
q1213  1805.962 10 1825.962 1864.541 1826.611
q1214  1809.364 10 1829.364 1867.943 1830.013
q215   1827.604  7 1841.604 1868.609 1841.931
q2136  1821.847  8 1837.847 1868.711 1838.270
q1135  1821.860  8 1837.860 1868.723 1838.282
q1125  1821.971  8 1837.971 1868.835 1838.394
q1136  1822.003  8 1838.003 1868.867 1838.426
q2135  1822.046  8 1838.046 1868.910 1838.468
q1137  1822.047  8 1838.047 1868.910 1838.469
q2137  1822.050  8 1838.050 1868.913 1838.472
q1126  1822.126  8 1838.126 1868.989 1838.548
q217   1828.122  7 1842.122 1869.127 1842.449
q1216  1810.576 10 1830.576 1869.155 1831.225
q1215  1810.603 10 1830.603 1869.182 1831.252
```

```
q1127  1822.409  8 1838.409 1869.273 1838.831
q11236 1816.582  9 1834.582 1869.303 1835.111
q1217  1810.855 10 1830.855 1869.434 1831.504
q11237 1816.802  9 1834.802 1869.524 1835.332
q12136 1805.133 11 1827.133 1869.570 1827.914
q216   1828.668  7 1842.668 1869.674 1842.996
q12137 1805.439 11 1827.439 1869.877 1828.220
q12134 1805.677 11 1827.677 1870.114 1828.458
q12135 1805.962 11 1827.962 1870.399 1828.743
q12146 1808.539 11 1830.539 1872.976 1831.320
```

The time taken for this call was 9 minutes and 31 seconds. Note that the simulation truth model ranks number 4 according to BIC of the models that have been fit.

7.6.2 Model 2

```
> options(show.error.messages = FALSE, warn = -1)
> try(load("b2out7a.rda"))
> options(show.error.messages = TRUE, warn = 0)
> if (!exists("b2out7a")) {
+     b2out7a <- aster.leaps(pred, fam, data = redata,
+         nsplit = 7, response = "resp2")
+     save(b2out7a, file = "b2out7a.rda")
+ }
> secs <- b2out7a$time[1]
> mins <- floor(secs/60)
> secs <- floor(secs - mins * 60)
> ilow <- b2out7a$fits[, "aic"] < b2out7a$fits[1, "aic"] +
+     10
> b2out7a$fits[ilow, , drop = FALSE]

               dev  p      aic      bic      cic
q1234671  199.5291 31 261.5291 381.1251 267.7681
q12346715 199.5179 32 263.5179 386.9718 270.1804
q12345671 187.0563 39 265.0563 415.5157 275.1209
q1234561  204.5308 31 266.5308 386.1268 272.7698
q123461   218.5492 24 266.5492 359.1396 270.2415
q1234617  217.8235 25 267.8235 364.2718 271.8358
q12345617 204.0389 32 268.0389 391.4928 274.7014
q1234615  218.5485 25 268.5485 364.9968 272.5608
q12346157 217.8075 26 269.8075 370.1138 274.1542
```

The call to this function took 16 minutes and 14 seconds. Note that the simulation truth model is not among the models under consideration (it depends on all 10 z_i).

```
> options(show.error.messages = FALSE, warn = -1)
> try(load("b2out7c.rda"))
```

```
> options(show.error.messages = TRUE, warn = 0)
> if (!exists("b2out7c")) {
+     b2out7c <- aster.leaps(pred, fam, data = redata,
+         nsplit = 7, response = "resp2", type = "AICc",
+         envir = b2out7a$envir)
+     save(b2out7c, file = "b2out7c.rda")
+ }
> secs <- b2out7c$time[1]
> mins <- floor(secs/60)
> secs <- floor(secs - mins * 60)
> ilow <- b2out7c$fits[, "cic"] < b2out7c$fits[1, "cic"] +
+     10
> b2out7c$fits[ilow, , drop = FALSE]
```

	dev	p	aic	bic	cic
q1234671	199.5291	31	261.5291	381.1251	267.7681
q12346715	199.5179	32	263.5179	386.9718	270.1804
q123461	218.5492	24	266.5492	359.1396	270.2415
q1234617	217.8235	25	267.8235	364.2718	271.8358
q1234615	218.5485	25	268.5485	364.9968	272.5608
q1234561	204.5308	31	266.5308	386.1268	272.7698
q12346157	217.8075	26	269.8075	370.1138	274.1542
q12345617	204.0389	32	268.0389	391.4928	274.7014
q12345671	187.0563	39	265.0563	415.5157	275.1209
q123471	223.8307	24	271.8307	364.4211	275.5230
q123671	225.5405	24	273.5405	366.1309	277.2328
q1234716	223.4512	25	273.4512	369.8995	277.4635
q1234715	223.6372	25	273.6372	370.0855	277.6495

The call to this function took 2 minutes and 25 seconds. Note that the simulation truth model is not among the models under consideration (it depends on all 10 z_i).

```
> options(show.error.messages = FALSE, warn = -1)
> try(load("b2out7b.rda"))
> options(show.error.messages = TRUE, warn = 0)
> if (!exists("b2out7b")) {
+     b2out7b <- aster.leaps(pred, fam, data = redata,
+         nsplit = 7, response = "resp2", type = "BIC",
+         envir = b2out7c$envir)
+     save(b2out7b, file = "b2out7b.rda")
+ }
> secs <- b2out7b$time[1]
> mins <- floor(secs/60)
> secs <- floor(secs - mins * 60)
> ilow <- b2out7b$fits[, "bic"] < b2out7b$fits[1, "bic"] +
+     10
> b2out7b$fits[ilow, , drop = FALSE]
```

```
            dev  p      aic      bic      cic
q12341  248.3135 18 284.3135 353.7563 286.3800
q123417 247.1159 19 285.1159 358.4166 287.4189
q123416 247.7541 19 285.7541 359.0549 288.0572
q123461 218.5492 24 266.5492 359.1396 270.2415
q123415 248.2524 19 286.2524 359.5531 288.5554
q12371  256.1983 18 292.1983 361.6411 294.2648
```

The time taken for this call was 19 minutes and 5 seconds. Note that the simulation truth model is not among the models under consideration (it depends on all 10 z_i).

7.7 Total Time

The total time taken for all calls to `aster.leaps` is

```
> leapout <- ls(pattern = "^b[12]out")
> secs <- 0
> for (i in seq(along = leapout)) {
+     bxout <- get(leapout[i])
+     secs <- secs + bxout$time[1]
+ }
> mins <- floor(secs/60)
> secs <- floor(secs - mins * 60)
> hrs <- floor(mins/60)
> mins <- floor(mins - hrs * 60)
```

1 hours, 43 minutes, and 43 seconds.

8 Frequentist Model Averaging

We do not know if any model under consideration (in the class $\mathcal{M}$) is correct; in our second example none are. Even if some model under consideration is correct, we do not know which one it is; in our first example neither AIC, AICc nor BIC ever selected the correct model. Thus it makes no sense to proceed as if the model selected by AIC, AICc or BIC is correct. So what do we do?

A recent proposal is frequentist model averaging (FMA), which copies Bayesian model averaging (BMA) in operation though not in philosophy. See Burnham and Anderson (2002) and Hjort and Claeskens (2003) for FMA and Hoeting et al. (1999) for BMA.

BMA does the Right Thing from the Bayesian point of view. A Bayesian considers everything unknown a parameter and formulates uncertainty about it as a prior distribution. Here both models and what the frequentist calls parameters within models are unknown. Given a prior on both models and parameters within models, the Bayesian computes the posterior distribution (on both models and parameters within models). Then, given something to predict (e. g., the fitness landscape), which is a function of model and parameter within model the Bayesian predicts this by averaging over the posterior distribution (over

both models and parameters within models). In this context full BMA would be very computationally intensive, requiring Markov chain Monte Carlo. FMA does roughly the same thing, is much easier to do, and perhaps works as well, although we do not investigate this.

In this section we will mostly follow Burnham and Anderson (2002, Chapter 4). The idea is to average inferences from various models that are reasonably well supported by the data. The virtue of averaging here is the same as everywhere else in statistics: it cancels errors more often than it amplifies them. In BMA, $\exp\bigl(-\frac{1}{2}\,\mathrm{BIC}(m)\bigr)$ is asymptotically proportional to the posterior probability of model m. Hence if $g(\theta)$ is a function of the parameters we are interested in estimating, the weighted average

$$\frac{\sum_{m\in\mathcal{M}} g(\hat{\theta}_m)\exp\bigl(-\frac{1}{2}\,\mathrm{BIC}(m)\bigr)}{\sum_{m\in\mathcal{M}}\exp\bigl(-\frac{1}{2}\,\mathrm{BIC}(m)\bigr)} \tag{8}$$

is an asymptotic approximation of the posterior expectation of $g(\theta)$. True BMA would use the true posterior probabilities and also integrate over θ within models (Hoeting et al., 1999), but this is a reasonable asymptotic approximation. Proceeding by analogy, frequentists would use

$$\frac{\sum_{m\in\mathcal{M}} g(\hat{\theta}_m)\exp\bigl(-\frac{1}{2}\,\mathrm{AIC}(m)\bigr)}{\sum_{m\in\mathcal{M}}\exp\bigl(-\frac{1}{2}\,\mathrm{AIC}(m)\bigr)} \tag{9}$$

or the same with AIC replaced by AICc (Burnham and Anderson, 2002, Chapter 4). There does not seem to be a strong theoretical justification for this particular form of weighted average (Hjort and Claeskens, 2003), but any averaging is better than no averaging.

Because we have not fit all models in $\mathcal{M}$ and do not want to (it would take a huge amount of time), we propose to use a short cut in the spirit of what Madigan and Raftery (1994) called "Occam's window." Instead of averaging over all the models under consideration, we average only over a random subset $\mathcal{A}$, which we define by

$$\mathcal{A} = \left\{ m \in \mathcal{M} : \mathrm{AIC}(m) < 2\log(c) + \min_{m'\in\mathcal{M}} \mathrm{AIC}(m') \right\} \tag{10}$$

where c is a constant chosen by the investigators. Then we replace $\mathcal{M}$ by $\mathcal{A}$ in (8) and (9) giving

$$\frac{\sum_{m\in\mathcal{A}} g(\hat{\theta}_m)\exp\bigl(-\frac{1}{2}\,\mathrm{AIC}(m)\bigr)}{\sum_{m\in\mathcal{A}}\exp\bigl(-\frac{1}{2}\,\mathrm{AIC}(m)\bigr)} \tag{11}$$

or the same with AIC replaced by AICc or BIC. Madigan and Raftery (1994) used in their examples $c = 20$ "by analogy with the popular .05 cutoff for P values" but that "popular" choice is itself arbitrary and without any theoretical justification, hence so is using $c = 20$ in BMA or FMA. Nevertheless, we will use the same choice.

The full "Occam's window" proposal of Madigan and Raftery (1994) involved another kind of pruning using a smaller subset than the $\mathcal{A}$ defined here, but Raftery and coauthors seem to have dropped this second kind of pruning; Hoeting et al. (1999) call it "optional" and Volinsky, et al. (1997) do not even mention it.

The discussants of Hoeting et al. (1999) were dubious about Occam's window and in reply Hoeting et al. (1999) admitted "we know of no formal theoretical support for it." It is merely a computational convenience. In FMA the change from (9) to (11) is, perhaps, a

bit less dubious, not because there is stronger justification for the pruning (10) but because there is less theoretical justification for (9), there being no philosophically ideal form of frequentist model averaging.

For our examples of FMA we will stop using AIC and use only AICc and BIC.

8.1 Five Predictors

8.1.1 Model 1

First we use the data simulated from Model 1. Since we now want to be sure we have all models in the set (10), we need to rerun `aster.leaps` using its `cutoff` argument.

```
> options(show.error.messages = FALSE, warn = -1)
> try(load("f1out5c.rda"))
> options(show.error.messages = TRUE, warn = 0)
> cutoff <- 2 * log(20)
> if (!exists("f1out5c")) {
+     f1out5c <- aster.leaps(pred, fam, data = redata,
+         nsplit = 5, response = "resp1", type = "AICc",
+         envir = b1out5b$envir, cutoff = cutoff)
+     save(f1out5c, file = "f1out5c.rda")
+ }
> secs <- f1out5c$time[1]
> mins <- floor(secs/60)
> secs <- floor(secs - mins * 60)
```

The call to this function took 0 minutes and 10 seconds. We had to fit 2 new models that we had not needed to fit in finding the minimum AIC, AICc and BIC.

```
> ilow <- f1out5c$fits[, "cic"] < f1out5c$fits[1, "cic"] +
+     cutoff
> mods <- f1out5c$fits[ilow, , drop = FALSE]
> print(mods)
```

```
             dev  p      aic      bic      cic
q1213    1805.962 10 1825.962 1864.541 1826.611
q12136   1805.133 11 1827.133 1869.570 1827.914
q12137   1805.439 11 1827.439 1869.877 1828.220
q12134   1805.677 11 1827.677 1870.114 1828.458
q12135   1805.962 11 1827.962 1870.399 1828.743
q12713   1800.006 14 1828.006 1882.017 1829.259
q121     1810.862  9 1828.862 1863.583 1829.391
q121346  1804.540 12 1828.540 1874.836 1829.466
q121367  1804.763 12 1828.763 1875.059 1829.689
q121347  1805.006 12 1829.006 1875.301 1829.932
q1214    1809.364 10 1829.364 1867.943 1830.013
q121356  1805.118 12 1829.118 1875.413 1830.043
q121357  1805.424 12 1829.424 1875.719 1830.350
```

```
q127136  1799.042 15 1829.042 1886.911 1830.479
q121345  1805.667 12 1829.667 1875.962 1830.593
q127134  1799.505 15 1829.505 1887.374 1830.942
q1213467 1804.026 13 1830.026 1880.179 1831.110
q1216    1810.576 10 1830.576 1869.155 1831.225
q1215    1810.603 10 1830.603 1869.182 1831.252
q12146   1808.539 11 1830.539 1872.976 1831.320
q1231    1804.280 13 1830.280 1880.433 1831.363
q127135  1800.005 15 1830.005 1887.874 1831.442
q1217    1810.855 10 1830.855 1869.434 1831.504
q1213456 1804.540 13 1830.540 1880.693 1831.623
q1271346 1798.142 16 1830.142 1891.869 1831.776
q1213567 1804.710 13 1830.710 1880.863 1831.793
q1271    1804.718 13 1830.718 1880.871 1831.802
q12371   1793.907 18 1829.907 1899.349 1831.973
q12147   1809.215 11 1831.215 1873.652 1831.996
q12145   1809.303 11 1831.303 1873.740 1832.084
q1213457 1805.004 13 1831.004 1881.157 1832.088
q12714   1803.051 14 1831.051 1885.062 1832.305
```

There are 32 models to average over (shown above).

For the "parameter" to average, we take the whole fitness surface. Since this is a function, we cannot fit it at all points. We will fit it at the 350 data points. First we define a function that takes the mean value parameter, τ in the notation in Geyer et al. (2007), to fitness; it uses the global variable `vars` defined on p. 4.

```
> tau2fit <- function(tau) {
+     tau <- matrix(tau, ncol = length(vars))
+     widx <- grep("^w[0-9]", vars)
+     apply(tau[, widx], 1, sum)
+ }
```

Thus, for example, the "simulation truth" fitness for Model 1 is given by

```
> true <- tau2fit(redata$tau1)
```

Since we did not save the whole fit, just the AIC, AICc, and BIC values for the models we fit during execution of the branch and bound algorithm, we need to refit the model in the set $\mathcal{A}$. Since we only know these models by their "names," which are character strings

```
> modnames <- dimnames(mods)[[1]]
> modnames
```

```
 [1] "q1213"    "q12136"   "q12137"   "q12134"   "q12135"
 [6] "q12713"   "q121"     "q121346"  "q121367"  "q121347"
[11] "q1214"    "q121356"  "q121357"  "q127136"  "q121345"
[16] "q127134"  "q1213467" "q1216"    "q1215"    "q12146"
[21] "q1231"    "q127135"  "q1217"    "q1213456" "q1271346"
```

```
[26] "q1213567" "q1271"    "q12371"   "q12147"   "q12145"
[31] "q1213457" "q12714"
```

we need a function that turns such strings into model formulas and fits the models. Such a function, called `redomod`, is in the file sourced on p. 31 that also contained the `aster.leaps` function. Here is how it is called.

```
> args(redomod)

function (string, data, response = "resp")
NULL
```

The `string` argument is the model specification, e. g., `"q1213"`, the `data` argument is the data frame containing the variables, for us always `redata`, and the `response` argument is the name of the response, for us either `"resp1"` or `"resp2"`.

Thus, for example, the fitness predicted by the model "selected" by AICc (the model with smallest AICc) is given by

```
> out <- redomod(modnames[1], data = redata, response = "resp1")
> wslct <- tau2fit(predict(out))
```

and the fitness predicted by the correct model, which we know because the data are simulated but which we would not know in real life, is given by

```
> out <- redomod("q12l", data = redata, response = "resp1")
> wbest <- tau2fit(predict(out))
```

Now we want to calculate the fitness predicted using FMA. First we calculate the weights used in the weighted average, then we refit the models and average the predictions of the models using these weights.

```
> wgt <- mods[, "cic"]
> wgt <- wgt - wgt[1]
> wgt <- exp(-wgt/2)
> wgt <- wgt/sum(wgt)
> wgt
```

```
      q1213      q12136      q12137      q12134      q12135
0.164478545 0.085754607 0.073559640 0.065327573 0.056649108
     q12713        q121     q121346     q121367     q121347
0.043757635 0.040967134 0.039459642 0.035292966 0.031262020
      q1214     q121356     q121357     q127136     q121345
0.030022122 0.029567238 0.025363385 0.023780430 0.022465986
    q127134    q1213467       q1216       q1215      q12146
0.018864697 0.017348325 0.016376809 0.016158695 0.015618748
      q1231     q127135       q1217    q1213456    q1271346
0.015285008 0.014692956 0.014244120 0.013422020 0.012434556
   q1213567       q1271      q12371      q12147      q12145
0.012325695 0.012274307 0.011265911 0.011137470 0.010658022
   q1213457      q12714
0.010638980 0.009545654
```

```
> wpred <- 0 * true
> modnames <- dimnames(mods)[[1]]
> for (i in seq(along = modnames)) {
+     out <- redomod(modnames[i], data = redata, response = "resp1")
+     wpred <- wpred + wgt[i] * tau2fit(predict(out))
+ }
```

Now we compare these "predictions" (actually, *estimates* is the better term in this situation). There are several criteria we could use for comparison. The most natural to statisticians is root-mean-square (RMS) error

```
> sqrt(mean((wslct - true)^2))

[1] 1.330927

> sqrt(mean((wpred - true)^2))

[1] 1.226865

> sqrt(mean((wbest - true)^2))

[1] 0.8031567
```

We see that FMA does slightly better than merely using the model "selected" by AICc, but neither does as well as using the correct model, which in real life we would not know.

Or we could use mean absolute error.

```
> mean(abs(wslct - true))

[1] 1.037055

> mean(abs(wpred - true))

[1] 0.963585

> mean(abs(wbest - true))

[1] 0.6575415
```

Same story.

Or we could use maximum error.

```
> max(abs(wslct - true))

[1] 4.297355

> max(abs(wpred - true))

[1] 4.525463
```

```
> max(abs(wbest - true))

[1] 1.761091
```

Same story. Since we get the same results by all three criteria, from now on we will only use RMS error.

We save these results for future reference.

```
> m1save <- data.frame(npred = c(5, 5, 2), type = c("select-AICc",
+     "FMA-AICc", "correct"), rms = c(sqrt(mean((wslct -
+     true)^2)), sqrt(mean((wpred - true)^2)), sqrt(mean((wbest -
+     true)^2))))
> print(m1save)

  npred        type       rms
1     5 select-AICc 1.3309273
2     5    FMA-AICc 1.2268651
3     2     correct 0.8031567
```

Now we redo everything using BIC instead of AICc.

```
> options(show.error.messages = FALSE, warn = -1)
> try(load("f1out5b.rda"))
> options(show.error.messages = TRUE, warn = 0)
> cutoff <- 2 * log(20)
> if (!exists("f1out5b")) {
+     f1out5b <- aster.leaps(pred, fam, data = redata,
+         nsplit = 5, response = "resp1", type = "BIC",
+         envir = f1out5c$envir, cutoff = cutoff)
+     save(f1out5b, file = "f1out5b.rda")
+ }
> secs <- f1out5b$time[1]
> mins <- floor(secs/60)
> secs <- floor(secs - mins * 60)
```

The call to this function took 1 minutes and 57 seconds. We had to fit 55 new models that we had not yet needed to fit.

```
> ilow <- f1out5b$fits[, "bic"] < f1out5b$fits[1, "bic"] +
+     cutoff
> mods <- f1out5b$fits[ilow, , drop = FALSE]
> print(mods)

          dev  p      aic      bic      cic
q213 1822.091  7 1836.091 1863.097 1836.419
q113 1822.145  7 1836.145 1863.151 1836.473
q112 1822.442  7 1836.442 1863.448 1836.770
q121 1810.862  9 1828.862 1863.583 1829.391
```

```
q21   1828.689  6 1840.689 1863.836 1840.933
q1123 1817.493  8 1833.493 1864.356 1833.915
q1213 1805.962 10 1825.962 1864.541 1826.611
q211  1825.842  7 1839.842 1866.848 1840.170
q11   1831.884  6 1843.884 1867.032 1844.129
q214  1826.384  7 1840.384 1867.390 1840.712
q123  1832.756  6 1844.756 1867.904 1845.001
q1214 1809.364 10 1829.364 1867.943 1830.013
q2113 1821.108  8 1837.108 1867.972 1837.531
q1124 1821.207  8 1837.207 1868.070 1837.629
q1134 1821.225  8 1837.225 1868.089 1837.647
q114  1827.196  7 1841.196 1868.202 1841.524
q12   1839.267  5 1849.267 1868.556 1849.441
q2134 1821.706  8 1837.706 1868.569 1838.128
q215  1827.604  7 1841.604 1868.609 1841.931
q2136 1821.847  8 1837.847 1868.711 1838.270
q1135 1821.860  8 1837.860 1868.723 1838.282
q1125 1821.971  8 1837.971 1868.835 1838.394
q1136 1822.003  8 1838.003 1868.867 1838.426
q2135 1822.046  8 1838.046 1868.910 1838.468
q1137 1822.047  8 1838.047 1868.910 1838.469
q2137 1822.050  8 1838.050 1868.913 1838.472
q1126 1822.126  8 1838.126 1868.989 1838.548
```

There are 27 models to average over (shown above) which have "names"

```
> modnames <- dimnames(mods)[[1]]
> modnames

 [1] "q213"  "q113"  "q112"  "q121"  "q21"   "q1123" "q1213"
 [8] "q211"  "q11"   "q214"  "q123"  "q1214" "q2113" "q1124"
[15] "q1134" "q114"  "q12"   "q2134" "q215"  "q2136" "q1135"
[22] "q1125" "q1136" "q2135" "q1137" "q2137" "q1126"
```

The fitness predicted by the model "selected" by BIC is given by

```
> out <- redomod(modnames[1], data = redata, response = "resp1")
> wslct <- tau2fit(predict(out))
```

We do not have to redo `wbest`. It is the same as before (it did not depend on whether we were using AICc or BIC).

Now we want to calculate the fitness predicted using FMA. First we calculate the weights used in the weighted average, then we refit the models and average the predictions of the models using these weights.

```
> wgt <- mods[, "bic"]
> wgt <- wgt - wgt[1]
> wgt <- exp(-wgt/2)
```

```
> wgt <- wgt/sum(wgt)
> wgt

      q213       q113       q112       q121        q21
0.145578985 0.141712034 0.122145425 0.114164324 0.100597739
     q1123       q1213        q211         q11        q214
0.077557677 0.070701031 0.022316010 0.020356630 0.017018561
      q123       q1214       q2113       q1124       q1134
0.013159961 0.012904996 0.012722073 0.012109435 0.012000111
      q114         q12       q2134        q215       q2136
0.011338569 0.009496791 0.009435310 0.009249056 0.008791822
     q1135       q1125       q1136       q2135       q1137
0.008736481 0.008263427 0.008132138 0.007959497 0.007958019
     q2137       q1126
0.007945509 0.007648389

> wpred <- 0 * true
> modnames <- dimnames(mods)[[1]]
> for (i in seq(along = modnames)) {
+     out <- redomod(modnames[i], data = redata, response = "resp1")
+     wpred <- wpred + wgt[i] * tau2fit(predict(out))
+ }
```

Now we compare these estimates using RMS error and also compare with our other estimates using AICc.

```
> tmp <- data.frame(npred = rep(5, 2), type = c("select-BIC",
+     "FMA-BIC"), rms = c(sqrt(mean((wslct - true)^2)),
+     sqrt(mean((wpred - true)^2))))
> m1save <- rbind(m1save, tmp)
> m1save <- m1save[rev(order(m1save[, "rms"])), ]
> print(m1save)

  npred        type       rms
4     5  select-BIC 2.0894708
1     5 select-AICc 1.3309273
5     5     FMA-BIC 1.2585405
2     5    FMA-AICc 1.2268651
3     2     correct 0.8031567
```

Despite the fact that BIC is supposed to do better than AICc in this situation (where one of the models under consideration is true and a rather small one of them), it actually does worse in this particular example, the problem being that the model it "selects" "q213" is too small.

8.1.2 Model 2

Now we use the data simulated from Model 2.

```
> options(show.error.messages = FALSE, warn = -1)
> try(load("f2out5c.rda"))
> options(show.error.messages = TRUE, warn = 0)
> cutoff <- 2 * log(20)
> if (!exists("f2out5c")) {
+     f2out5c <- aster.leaps(pred, fam, data = redata,
+         nsplit = 5, response = "resp2", type = "AICc",
+         envir = b2out5b$envir, cutoff = cutoff)
+     save(f2out5c, file = "f2out5c.rda")
+ }
> secs <- f2out5c$time[1]
> mins <- floor(secs/60)
> secs <- floor(secs - mins * 60)
```

The call to this function took 0 minutes and 0 seconds. We had to fit 0 new models that we had not needed to fit in finding the minimum AIC, AICc and BIC.

```
> ilow <- f2out5c$fits[, "cic"] < f2out5c$fits[1, "cic"] +
+     cutoff
> mods <- f2out5c$fits[ilow, , drop = FALSE]
> print(mods)
```

```
             dev  p      aic      bic      cic
q12341  248.3135 18 284.3135 353.7563 286.3800
q123451 235.9033 24 283.9033 376.4937 287.5956
q123415 248.2524 19 286.2524 359.5531 288.5554
```

There are 3 models to average over (shown above), which have "names"

```
> modnames <- dimnames(mods)[[1]]
> modnames
```

```
[1] "q12341"  "q123451" "q123415"
```

The "simulation truth" fitness for Model 2 is given by

```
> wtrue <- tau2fit(redata$tau2)
```

The fitness predicted by the model "selected" by AICc is given by

```
> out <- redomod(modnames[1], data = redata, response = "resp2")
> wslct <- tau2fit(predict(out))
```

and the fitness predicted by the correct model, which we know because the data are simulated but which we would not know in real life, is given by

```
> out <- redomod("q123456789Al", data = redata, response = "resp2")
> wbest <- tau2fit(predict(out))
```

Now we want to calculate the fitness predicted using FMA. First we calculate the weights used in the weighted average, then we refit the models and average the predictions of the models using these weights.

```
> wgt <- mods[, "cic"]
> wgt <- wgt - wgt[1]
> wgt <- exp(-wgt/2)
> wgt <- wgt/sum(wgt)
> wgt

   q12341    q123451    q123415
0.5314843 0.2894129 0.1791028

> wpred <- 0 * wtrue
> modnames <- dimnames(mods)[[1]]
> for (i in seq(along = modnames)) {
+     out <- redomod(modnames[i], data = redata, response = "resp2")
+     wpred <- wpred + wgt[i] * tau2fit(predict(out))
+ }
```

Now we compare these estimates using root-mean-square error.

```
> m2save <- data.frame(npred = c(5, 5, 10), type = c("select-AICc",
+     "FMA-AICc", "correct"), rms = c(sqrt(mean((wslct -
+     true)^2)), sqrt(mean((wpred - true)^2)), sqrt(mean((wbest -
+     true)^2))))
> m2save <- m2save[rev(order(m2save[, "rms"])), ]
> print(m2save)

  npred        type      rms
3    10     correct 13.10318
2     5    FMA-AICc 12.29017
1     5 select-AICc 12.28608
```

We see that FMA does slightly worse than merely using the model "selected" by AICc, and both do better than using the correct model, because the correct model just has too many parameters to estimate well with this amount of data.

Now we redo everything using BIC instead of AICc.

```
> options(show.error.messages = FALSE, warn = -1)
> try(load("f2out5b.rda"))
> options(show.error.messages = TRUE, warn = 0)
> cutoff <- 2 * log(20)
> if (!exists("f2out5b")) {
+     f2out5b <- aster.leaps(pred, fam, data = redata,
```

```
+         nsplit = 5, response = "resp2", type = "BIC",
+         envir = f2out5c$envir, cutoff = cutoff)
+     save(f2out5b, file = "f2out5b.rda")
+ }
> secs <- f2out5b$time[1]
> mins <- floor(secs/60)
> secs <- floor(secs - mins * 60)
```

The call to this function took 0 minutes and 36 seconds. We had to fit 8 new models that we had not yet needed to fit.

```
> ilow <- f2out5b$fits[, "bic"] < f2out5b$fits[1, "bic"] +
+     cutoff
> mods <- f2out5b$fits[ilow, , drop = FALSE]
> print(mods)

             dev  p      aic      bic      cic
q12341   248.3135 18 284.3135 353.7563 286.3800
q123415 248.2524 19 286.2524 359.5531 288.5554
```

There are 2 models to average over (shown above) which have "names"

```
> modnames <- dimnames(mods)[[1]]
> modnames

[1] "q12341"  "q123415"
```

The fitness predicted by the model "selected" by BIC is given by

```
> out <- redomod(modnames[1], data = redata, response = "resp2")
> wslct <- tau2fit(predict(out))
```

We do not have to redo `wbest`. It is the same as before (it did not depend on whether we were using AICc or BIC).

Now we want to calculate the fitness predicted using FMA. First we calculate the weights used in the weighted average, then we refit the models and average the predictions of the models using these weights.

```
> wgt <- mods[, "bic"]
> wgt <- wgt - wgt[1]
> wgt <- exp(-wgt/2)
> wgt <- wgt/sum(wgt)
> wgt

    q12341    q123415
0.94776716 0.05223284
```

```
> wpred <- 0 * wtrue
> modnames <- dimnames(mods)[[1]]
> for (i in seq(along = modnames)) {
+     out <- redomod(modnames[i], data = redata, response = "resp2")
+     wpred <- wpred + wgt[i] * tau2fit(predict(out))
+ }
```

Now we compare these estimates using root-mean-square error.

```
> tmp <- data.frame(npred = rep(5, 2), type = c("select-BIC",
+     "FMA-BIC"), rms = c(sqrt(mean((wslct - true)^2)),
+     sqrt(mean((wpred - true)^2))))
> m2save <- rbind(m2save, tmp)
> m2save <- m2save[rev(order(m2save[, "rms"])), ]
> print(m2save)

  npred        type      rms
3    10     correct 13.10318
2     5    FMA-AICc 12.29017
4     5  select-BIC 12.28608
1     5 select-AICc 12.28608
5     5     FMA-BIC 12.28605
```

The order between "selection" and FMA is now confused, with one FMA doing better than the "selection" and the other worse, but all of the model selection and model averaging estimators are better than using the correct model with 10 predictors and 69 parameters to estimate.

8.2 Summary

We originally intended to repeat the analyses in this section using more predictor variables, but since we have seen the pattern we expected (more or less) with five predictors, and since this model selection and model averaging are a minor point of the paper (although most of the work), we stop here.

9 Discussion

It is hard to know what lessons to draw from simulations. All simulation studies are designed, consciously or unconsciously, to "prove" a particular point. Since nearly any method can be made to look good if the simulation is chosen precisely to make it look good, simulations actually prove nothing.

All we can say about this simulation is that we had, or at least were consciously aware of, no ax to grind other than illustrating that principal components regression is a bad idea. But this latter point is so well understood by statisticians, that we did not bother to illustrate it with our simulations. It is so obvious that principal components would do a horrible job on our example, that there would be no point to actually illustrating this.

Among model selection and model averaging ideas that actually have some theoretical justification, we have no ax to grind, not being experts in the area. We tried some and they worked, more or less. Most readers, however, are probably disappointed in how well they worked. Many users of statistics have no idea how badly most model selection schemes work on realistic problems, and the actual performance of the best schemes known is much worse than users desire. But there is no magic, statistics works as well as it works. Short of a fairy godmother with a magic wand, it is clear that no model selection method known will do much better than the ones we tried here. We make no claim that the methods we tried are optimal; more complicated methods might do slightly better, but not much better.

Of course, how well methods work depend on how obvious the true model is. If the effects are made large enough or if the sample size is large enough, then any method that is any good (this does not include principal components regression) will select the correct model. If one increases the sample size `nind` (p. 3) or if one increases the strength of the quadratic effect `ascal` (p. 11), then the problem becomes much easier and for sufficiently large `nind` or `ascal` all of the methods we tried will "select" the simulation truth model with reasonably high probability. But how realistic is that? Note that the data we actually simulated is very easy when the true predictors are known (P-values on p. 18). Most scientists plan experiments just large enough to show something, not so large that everything is obvious without statistics.

We did, at least, show two contrasting situations. When the true model, which is, of course, unknown in real life, is "sparse" with only a few regression coefficients nonzero, then BIC does better. It has a bias towards small models, so when this bias works in its favor, it does better. When the true model, is non-sparse with many regression coefficients nonzero, then AIC and AICc do better. They have a bias towards large models, so when this bias works in their favor, they do better. Choose whichever you like, depending on your opinion about the true state of affairs. As we have repeatedly mentioned, Burnham and Anderson (2002, Section 1.2.5) are particularly emphatic about the biological unrealism of "true" (at least simulation truth) models with only a few parameters. Although we have nothing particular to add to this, we agree, for what it is worth. Thus we would usually use AICc rather than BIC.

Frequentist model averaging is new (less than ten years old) and we have even less to say about that. It does seem to work and work better than "selecting" a model and pretending it is true, a threadbare pretense when there are thousands of models under consideration and a negligible chance of selecting the correct one. In such circumstances, "selecting" a model and pretending it is true, especially overinterpreting the "selected" model and claiming that the predictors "selected" are the ones that explain the phenomena, is clearly wrong, and frequentist model averaging is a sensible substitute.

References

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. *Second International Symposium on Information Theory*, 267–281.

Burnham, K. P. and Anderson, D. R. (2002). *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*, 2nd ed. New York: Springer-Verlag.

Cook, R. D. (2007). Fisher lecture: Dimension reduction in regression (with discussion). *Statistical Science*, in press. `http://www.imstat.org/sts/future_papers.html`

Cox, D. R. (1968). Notes on some aspects of regression analysis. *Journal of the Royal Statistical Society, Ser. A*, **131**, 265–279.

Furnival, G. M. and Wilson, R. W., Jr. (1974). Regressions by Leaps and Bounds *Technometrics*, **16**, 499–511. Reprinted, *Technometrics*, **42**, 69–79.

Geyer, C. J. and Shaw, R. G. (2008). Supporting Data Analysis for a talk to be given at Evolution 2008 University of Minnesota, June 20–24. University of Minnesota School of Statistics Technical Report No. 669. `http://www.stat.umn.edu/geyer/aster/`

Geyer, C. J., Wagenius, S. and Shaw, R. G. (2007). Aster models for life history analysis. *Biometrika*, **94** 415–426.

Hand, D. J. (1981). Branch and bound in statistical data analysis. *The Statistician*, **30**, 1–13.

Hjort N. L. and Claeskens G. (2003). Frequentist model average estimators. *Journal of the American Statistical Association*, **98**, 879–899.

Hoeting, J. A., Madigan, D., Raftery, A. E., and Volinsky, C. T. (1999). Bayesian model averaging: A tutorial (with discussion). *Statistical Science*, **19**, 382–417. Corrected version available at `http://www.stat.washington.edu/www/research/online/1999/hoeting.pdf`.

Lande, R. and Arnold, S. J. (1983). The measurement of selection on correlated characters. *Evolution*, **37**, 1210–1226.

Madigan, D. and Raftery, A. E. (1994). Model selection and accounting for model uncertainty in graphical models using Occam's window. *Journal of the American Statistical Association*, **89**, 1535–1546.

R Development Core Team (2008). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. `http://www.R-project.org`.

Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, **6**, 461–464.

Shaw, R. G., Geyer, C. J., Wagenius, S., Hangelbroek, H. H., and Etterson, J. R. (2008). Unifying life history analysis for inference of fitness and population growth. *American Naturalist*, in press. `http://www.stat.umn.edu/geyer/aster/`

Shaw, R. G., Geyer, C. J., Wagenius, S., Hangelbroek, H. H., and Etterson, J. R. (2007). Supporting data analysis for "Unifying life history analysis for inference of fitness and population growth". University of Minnesota School of Statistics Technical Report No. 658 `http://www.stat.umn.edu/geyer/aster/`

Sugiura, N. (1978). Further analysis of the data by Akaike's information criterion and the finite corrections. *Communications in Statistics, Theory and Methods*, **A7**, 13–26.

Volinsky, C. T., Madigan, D., Raftery, A. E. and Kronmal, R. A. (1997). Bayesian model averaging in proportional hazard models: Assessing the risk of a stroke. *Applied Statistics*, **46**, 433–448.